On the structure of Bayesian network for Indonesian text document paraphrase identification

Ario Harry Prayogo, Mohamad Syahrul Mubarok, Adiwijaya
School of Computing, Telkom University
ariohp@telkomuniversity.ac.id, msyahrulmubarok@telkomuniversity.ac.id, adiwijaya@telkomuniversity.ac.id

Abstract. Paraphrase identification is an important process within natural language processing. The idea is to automatically recognize phrases that have different forms but contain same meanings. For example if we input query “causing fire hazard”, then the computer has to recognize this query that this query has same meaning as “the cause of fire hazard”. Paraphrasing is an activity that reveals the meaning of an expression, writing, or speech using different words or forms, especially to achieve greater clarity. In this research we will focus on classifying two Indonesian sentences whether it is a paraphrase to each other or not. There are four steps in this research, first is preprocessing, second is feature extraction, third is classifier building, and the last is performance evaluation. Preprocessing consists of tokenization, non-alphanumerical removal, and stemming. After preprocessing we will conduct feature extraction in order to build new features from given dataset. There are two kinds of features in the research, syntactic features and semantic features. Syntactic features consist of normalized levenshtein distance feature, term-frequency based cosine similarity feature, and LCS (Longest Common Subsequence) feature. Semantic features consist of Wu and Palmer feature and Shortest Path Feature. We use Bayesian Networks as the method of training the classifier. Parameter estimation that we use is called MAP (Maximum A Posteriori). For structure learning of Bayesian Networks DAG (Directed Acyclic Graph), we use BDeu (Bayesian Dirichlet equivalent uniform) scoring function and for finding DAG with the best BDeu score, we use K2 algorithm. In evaluation step we perform cross-validation. The average result that we get from testing the classifier as follows: Precision 75.2%, Recall 76.5%, F1-Measure 75.8% and Accuracy 75.6%.

1. Introduction

Natural Language Processing (NLP) is a technique that is used to analyze and represent human language automatically. It learns mathematical and computational model from many aspects of the language. NLP is used to take grammatical structure and build an output based on rules that present in the language which is the object of the process [1].

There are examples of NLP implementation such as plagiarism detection, information retrieval, text summarization, question answering, and machine translation. There is one essential process for plagiarism detection called paraphrase identification. Paraphrase is a rephrase of a writing, speech, or an expression of a level in language form to another form without altering its meaning; on another word paraphrase can be translated as a different expression of a text to another form to reveal the implicit meaning [2]. There is research regarding paraphrase identification of a social media (twitter).
There’s one thing that makes paraphrase identification is important. If we are doing plagiarism detection, then we need to make an automated system that can determine phrases that are different but having same meaning. For example there are pair of phrase, first phrase is “the source of wildfire” and the second phrase is “cause of wildfire”. The system must be able to identify the pair of phrase is a paraphrase. In Indonesian Language structure there are prefix, suffix, infix and circumfix which making NLP more complex than English.

To solve the problem, we try a method that consists several steps, the first is preprocessing, second is feature extraction, third is classifier building, and the last is performance evaluation. Preprocessing step is to improve data quality and remove unnecessary computation by reducing data noise. There are four steps in preprocessing that are tokenization, non-alphanumerical removal, stemming and translation. We use Nazief-Adriani algorithm for stemming because it gives the best result for Indonesian Language corpus [3]. After preprocessing we perform feature extraction process. The idea of using the process is to build new features from dataset. The first kind of extracted features is syntactic features which consist of normalized levenshtein distance feature, term-frequency (tf) based cosine similarity feature, and LCS (Longest Common Subsequence) feature. The second kind of feature is semantic features which calculate similarity between term based on distance in semantic tree (in our case we use WordNet). The semantic distance features consist of Wu and Palmer Features and Shortest Path Features. The next phase of paraphrase identification is classification using Bayesian Networks. Parameter that is used in the classifier is Maximum A Posteriori (MAP). For structure learning of Bayesian Networks DAG (Directed Acyclic Graph), we use BDeu (Bayesian Dirichlet equivalent uniform) scoring function and for finding DAG with the best BDeu score, we use K2 algorithm. In evaluation step we perform n-fold cross-validation with the number of n is 20.

Bayesian Networks is a probabilistic based modeling method that represent a variable set and the conditional dependency through a Directed Acyclic Graph (DAG) [4] [5]. There are four reasons why we choose Bayesian Networks as the classifier, first Bayesian Networks can handle incomplete dataset, second it allows the learning process based on cause and effect, third it is based on Bayesian Statistics which facilitate combination between data and knowledge domain, and fourth it provide a way to avoid overfit data [6].

2. Related Works
There are some researches regarding paraphrase identification such as “PDLK: Plagiarism Detection Using Linguistic Knowledge” [7]. The research proposes a method to identify paraphrase by combining semantic similarity along with word order similarity. The idea is to get a new feature that can handle similarity of a document semantically. For a limit value to separate document whether the content is same or not, they attempt an observation to get a threshold value and alfa weighting. The best alfa weighting value is 0.8 and the best threshold value is 0.6 which produce performance evaluation score of 73.9%, the evaluation uses F1-Measure approach. The research is related to ours in way that the research uses similar method for handling the similarity of a document (in our case is phrase or sentence). The difference of their research with ours is their research takes word order into account whilst our research uses two methods of semantic similarity instead and combined them with syntactic features.

Another research that discuss about paraphrase identification is using translated source where the translated language result is confirmed having same value with original document [8]. The research performs paraphrase identification by translating the phrases into desired language. They later perform an unsupervised learning algorithm for paraphrase extraction. They claim that the approach has the ability to extract multiple kind of paraphrases by doing identification of syntactic paraphrases and extraction of morpho-syntactic paraphrasing. There are other researches such as the research which uses named entity anchors [9]. The research main purpose is to automatically extract paraphrases from a corpus which is from headline news articles. They use named entity and POS Tagging to extract pattern from news article [10]. There is research about text classification using POS Tagging to measure polarity of words [11].
Paraphrase Identification and semantic similarity in twitter with simple features [12] perform syntactic and string similarity between pair of sentences by using logistic regression model with 18 features based on N-Grams. They use Edit Distance between sentences as a feature and normalize the score on the number of token of the shortest sentence. They claim that the feature configuration is best in term of performance score during evaluation on training and development data. In classification process they used several classification methods which Voted Perceptron method returns the highest F1-score of them all with F1-score reaching 74.6%.

Paraphrase identification in Indonesian Corpus [13] tried to identify Indonesian sentence pair whether it’s a paraphrase or not. The research uses levenstein distance and wu palmer method as features (nodes). The total nodes number is three including the class, then the nodes are used in classifier Bayesian Networks Classifier. For Bayesian Networks structure, the research specifies all possible combination of nodes, resulting 25 DAG structures. The research performance evaluation using F1-score which is 71.5%.

3. Research Method

In this section we describe our paraphrase identification method. The general process that is performed in this research is described in Figure 1.

![Figure 1 Research Method of Paraphrase Identification](image-url)
The research method includes four main processes. In the first step, preprocessing data with tokenization, non-alphanumeric removal, stemming, and translation. The second step is feature extraction which is divided into two types of features, semantic and syntactic. The semantic features consist of Wu and Palmer feature, and Shortest Path feature. The syntactic features consist of normalized levenshtein distance feature, LCS feature, and Tf based cosine similarity feature. All of it means that the number of features used in this research are five features.

Those five features will be used in the third step which is classifier building. Third step is finding the best DAG based on highest number of BDeu score using K2 Algorithm. The fourth step is classifier building using the DAG that is returned from using K2 Algorithm. The detail of each process described in Figure 1 is explained as follows.

3.1 Building Dataset

In this process, we are collecting and building paraphrase dataset of Indonesian Language. The data consists of 1004 pair of phrases/sentences along with its label either 1 which is a paraphrase or 0 which is not a paraphrase. This dataset has balance class, it means the number of dataset which is paraphrase is equal with the number of dataset which is not a paraphrase.

When we collect the data, we use several sources which mainly from Indonesian online news articles. We collect all the news from several news websites with similar article. For example, we took news articles from Merdeka¹ and Republika². After we found pair of articles with similar theme, we try to manually find sentences in both articles that is having similar meaning. After that the pair of sentences are compiled as data in the dataset. There are other minor amounts of sources that are used as paraphrase data such as local Indonesian course text book and daily phrase/sentence that usually found in twitter. All of that is just to produce correct paraphrase, for false/incorrect paraphrase we get help from Indonesian Language experts to generate non-paraphrase data. The last phase of building dataset is validation with Indonesian Language experts to identify class/label whether the label is correct or incorrect, if the data is incorrect the experts will change the label to the correct label. Explanation of dataset distribution is explained in Table 1.

| Class          | Number of Data | Percentage |
|----------------|----------------|------------|
| Paraphrase     | 502            | 50%        |
| Non-paraphrase | 502            | 50%        |
| Total Dataset  | 1004           | 100%       |

3.2 Preprocessing

This process is the start of processing data before we identify whether a pair phrase is a paraphrase or not. The purpose of this step is to improve data quality and remove unnecessary computation by reducing data noise, all of that is to pursue performance improvement of the classifier. This preprocessing consists of four steps which will be explained below.

3.2.1 Tokenization, the process that convert phrase or sentences into vector of words. The separation is necessary for stemming process.

3.2.2 Non-alphanumeric removal, the process that remove all characters except numbers and letters. By doing this process, the unnecessary information is removed. For example when we perform semantic feature extraction, all characters except numbers and letter will not giving any improvement on the feature quality. That because in this research, semantic feature extraction uses WordNet which only support words. So, if the non-alphanumeric

¹ https://www.merdeka.com/uang/ini-kemampuan-pesawat-n219-buatan-pt-di.html
² http://nasional.republika.co.id/berita/nasional/daerah/17/08/16/ouro87377-ini-keunggulan-pesawat-n219-buatan-pt-dirgantara-indonesia
Characters will not be useful then we don’t need to include those characters into computation in the first place. We need to mention that non-alphanumeric character like “,” and “.” will not be removed if the character represents decimal number.

3.2.3 **Stemming**, the idea of this process is getting root word from available words in the dataset. The goal of getting root word is to prevent calculation mistake when performing syntactic feature extraction. For example when pair sentences are having same root word for all their words but only different on affix, suffix, infix or circumfix, if we do not perform stemming, the distance score of syntactic feature will not be 0, but if we perform the stemming we will get distance score of 0. If we get score 0 it means the meaning of pair sentence is equal, other than that it means the pair has different meaning whether it’s little bit different or explicitly different.

3.2.4 **Translation**, we attempt to translate the vector data into English using Google translator API. The reason why we perform translation because the lack of completed Indonesian thesaurus that is available in the moment. The current Indonesian thesaurus is still on development and not ready to be used as semantic similarity source.

3.3 **Feature Extraction**
In this process we will extract new features from set of data that is resulted by preprocessing. There are two feature that we will extract which are syntactic feature and semantic feature. Based on [12] that we can measure document similarity using both features. Before we explain each feature in feature extraction we like to explain a vector normalization method called cosine similarity or semantic-vector approach [14] to measure the similarity between sentences. The method is created for each two phrases/sentences, it means each phrase/sentence will have the vector. The pair vectors is containing words from pair phrases/sentences. The equation to calculate sentence similarity as follows:

\[
Sim_{\text{cosine}}(s_1, s_2) = \frac{\sum_{i=1}^{m} (w_1i - w_2i)}{\sqrt{\sum_{i=1}^{m} w_1i^2} \times \sqrt{\sum_{i=1}^{m} w_2i^2}}
\]  

Where \(S_1 = (w_{11}, w_{12}, ..., w_{1n})\) and \(S_2 = (w_{21}, w_{22}, ..., w_{2n})\) are the vectors of sentence \(S_1\) and \(S_2\). The \(w_{jk}\) is the weight score of the \(j^{th}\) word in vector \(S_k\), \(n\) is the number of words in the vector.

3.3.1 **Semantic Feature**. The semantic feature is a feature that calculate semantic distance of pair phrase/sentence. The goal is to determine similarity of the pair by using WordNet as the semantic tree. Semantic feature is containing two features which are Wu Palmer feature and Shortest Path feature.

a. Wu Palmer
WUP (Wu and Palmer) is our first semantic feature, the method is very simple and having good trade-off between performance and execution time cost [15]. WUP method determines similarity by calculating the depths between two synsets in the Wordnet taxonomies, also with the depth of LCS. The equation of WUP method is as follows [16].

\[
Sim_{\text{wup}}(s_1, s_2) = \frac{2 \times \text{depth}(\text{lcs})}{(\text{depth}(s_1) + \text{depth}(s_2))}
\]

For depth(lcs) denotes the depth of the lcs of sentence. LCS itself means the least common subsume.
b. Shortest Path
This method calculates shortest path between pair of term and return minimum distance between them. It based on shortest path that connects the senses in the is-a (hyponym/hyponym) taxonomy. Most of the links in Wordnet come in pairs like hyponym-hyponym, antonym-antonym and meronym-meronym. Most of derivational links are bidirectional which means part of Wordnet is a large undirected graph [17]. A problem of calculating semantic similarity between two words can be translated into shortest path problem of undirected graph G(V,E). It means we can apply basic searching algorithm such as Bidirectional Breadth First Search. Edge length is defined by equation as follows.

$$
\text{dist}(S,T) \leq \min \text{path\_length}(s,t)
$$

Where s is source term and t is target term, and s ∈ S and t ∈ T.

c. Calculating similarity with semantic-vector method
This process is to calculate sentence similarity with semantic-vector approach [7]. The following steps are performed to calculate the semantic similarity.

i. Creating semantic vector
Each Semantic-vector is created by words from each phrase/sentence. Each value in vector corresponds to a word in phrase/sentence, it means the dimension of the vector equals the number of words that are contained in a phrase/sentence $S_1$ or $S_2$. For $S_1 = \{t_{11}, t_{12},...,t_{1n}\}$ and $S_2 = \{t_{11}, t_{12},...,t_{1n}\}$ where $t_{ij}$ is term/word in sentence i and word j.

ii. Adding semantic weight to vector
Each row of semantic-vector is weighted using semantic similarity value between words from word set of sentences. There some rules for adding weight to the vector:

1. If the word, $w$ from sentence $S_1$ appear in the sentence $S_2$, the weight score for $w$ in the semantic-vector is set to 1. Else, go to next step.
2. If sentence $S_2$ does not contain $w$, then calculate similarity score using either Wu and Palmer or shortest distance method. The result of this calculation is a array represents similarity measure between $w$ and each word in $S_2$. It means the array dimension is equal with the number of words contained in $S_2$.
3. If the array is not empty then we set the weight score for $w$ as maximum value in array result. If array is empty then the weight score for $w$ is set to 0.

iii. After we acquire semantic-vector weight, we compute the similarity between $S_1$ and $S_2$ using equation 1.

3.3.2 Syntactic Feature. Syntactic feature is a method which calculates distance of pair phrase/sentence based on lexical characteristics of a sentence. The syntactic feature consists of three features they are normalized levenshtein distance, lcs and tf based cosine similarity.

a. Normalized Levenshtein distance
The distance calculation is using Normalized Levenshtein Distance method. We use this method because this method able to handle addition, reduction, or changes of a letter in the phrase/sentence. The result of syntactic feature is a number that range between 0 and 1, where 0 means the phrase/sentence is totally equal and 1 means totally different. There is three operations that this method takes into account. The explanation of the operations as follows:

i. Character addition, this operation basically adds a character into a word. For example in Indonesian, word “asa” becomes “asap” where there’s an addition of “a” character p.

ii. Character removal, this operation removes a character from given word. For example in Indonesian, word “makan” if we remove first character of the word, then the word becomes “akan”
iii. Character substitution, this operation changes a character from given word with another character. For example word “angan” if we change character “a” in digit 4 with character “i” then the word becomes “angin” [18].

The Levenhstein distance [18] gives the result of an edit distance between pair sentences. But the result is not enough because the value returned by the method depends on the length of the sentences. Which means shorter the sentences will likely return smaller value than longer sentences. To handle the problem, we use normalization, which divide the result with the length of the longest sentence between S₁ and S₂.

b. Term frequency based cosine similarity
   This feature main point is to consider the number of word appearance in a sentence. The steps performed in the process as follows.
   1. Counts term frequency for each word in a sentence, for both S₁ and S₂, the result is in vector form which represents the number of word appearances in a sentence. It means we will have two vectors.
   2. Calculate the distance between those vectors using cosine similarity in equation 1.
   The result of this process returns a value between 0 and 1, which is a distance in a form of cos value. If the result is 0, it means the pair sentences are completely different. If the result is 1, it means the pair sentences are completely the same.

c. Longest common subsequence (LCS)
   To determine similarity with this method we perform comparation between words from S₁ and S₂. Subsequence is any word that is acquired by removing zero or more symbols from w. Sub-word of w means that a subsequence of w that consists of consecutive symbols. For example ‘ABCBDAB’ and ‘BDCABA’ give us ‘BCBA’ as LCS result. A common subsequence of w and w'. We gain words similarity in the sentences by measuring length of LCS [19].
   After obtain LCS result for each word, we count the length of LCS for each result and insert all the number into a vector. This means that the vector contains number of LCS length that represent each word in the sentence. We do this for each sentence which means there are two vectors. Next step we compute similarity using cosine similarity in equation 1.

3.4 Building Classifier
   In this task we perform two processes, first is finding best DAG structure and the second is building classification. We search the best DAG structure using K2 algorithm with BDeu as scoring function in structure learning process. The BDeu score function is explained in equation 4 [20].

\[
BDeu(N : D, \alpha) = \sum_{i} \left\{ \sum_{j} \log \Gamma \left( \frac{\alpha}{q_{ij}} \right) - \log \Gamma \left( \frac{\alpha + n_{ij}}{q_{ij}} \right) + \sum_{k} \log \Gamma \left( \frac{\alpha}{r_{ijk} \cdot q_{ij}} \right) - \log \Gamma \left( \frac{\alpha}{r_{ijk} \cdot q_{ij}} \right) \right\} \tag{4}
\]

Where:
\( \alpha \) : an alpha value or hyperparameter
\( q_{ij} \) : instantiation number parent \( X_i \)
\( r_{ij} \) : Value combination of \( X_i \)
\( n_{ij} \) : The number of instance of \( X_i = j \)
\( n_{ijk} \): The number of instance of \( X_i = k \) and parent from \( i = j \)

The first process of building classifier is to find best DAG structure using K2 algorithm. The following pseudocode is the algorithm of K2 [21].

1. \( \text{procedure K2; } \)
2. \{Input: n, o, u, D\}  
3. \{Output: For each node, a printout of the parents of the node.\}  
4. \( \text{for } i := 1 \text{ to } n \text{ do } \)
5. \( \pi_i := \emptyset; \)
6. \( P_{\text{old}} := g(i, \pi_i); \) \( \text{This function is computed using equation (4).} \)  
7. \( \text{OKToProceed := true} \)
8. \( \text{while OKToProceed and } |\pi_i| < u \text{ do } \)
9. \( \text{let } z \text{ be the node in } \text{Pred}(x_i) - \pi, \text{ that maximizes } g(i, \pi_i \cup \{z\}); \)
10. \( P_{\text{new}} := g(i, \pi_i \cup \{z\}); \)
11. \( \text{if } P_{\text{new}} > P_{\text{old}} \text{ then } \)
12. \( P_{\text{old}} := P_{\text{new}}; \)
13. \( \pi_i := \pi_i \cup \{z\} \)
14. \( \text{else OKToProceed := false;} \)
15. \( \text{end } \{\text{while}\}; \)
16. \( \text{write(`Node:', } x_i, \text{ `Parents:', } \pi_i) \)
17. \( \text{end } \{\text{for}\}; \)
18. \( \text{end } \{\text{K2}\}; \)

Where:
\( n \) = A set of n nodes in graph,  
\( o \) = An ordering on the nodes, in our case the order represents naive bayes form,  
\( u \) = The number of parents a node may have,  
\( D \) = A database containing m cases

After obtain DAG structure, we create joint probability for the graph. The function for joint probability represented in the function 5 as follows.

\[
P(U) = \prod_{i=1}^{n} P(p_k|\text{parents}(p_k)) \quad (5)
\]

For parameter learning we use Maximum A Posteriori (MAP) that is explained in function 6.

\[
\theta_{ijk} = \frac{\alpha_{ijk} + n_{ijk}}{\sum_k(\alpha_{ijk} + n_{ijk})} \quad (6)
\]
Where:

\[ \theta_{ijk} = \text{Parameter where } X_i = k \text{ and Parent of } i = j \]

\[ n_{ijk} = \text{The number of instance of } X_i = k \text{ and parent from } i = j \]

\[ \alpha_{ijk} = \text{Smoothing} \]

The smoothing value can be obtained through function 7.

\[ \alpha_{ijk} = \frac{\alpha}{r_i \cdot q_i} \]  \hspace{1cm} (7)

Where:

\[ \alpha_{ijk} = \text{Smoothing} \]

\[ \alpha = \text{an alpha value or hyperparameter} \]

\[ r_i = \text{value combination of } X_i \]

\[ q_i = \text{Instantiation parent number of } X_i \]

Alpha (\( \alpha \)) value is used to prevent dividing by zero, in our case we use \( \alpha = 0.1 \).

4. Result and Evaluation

In this section we will evaluate the result from our work, first we will evaluate the combination of the features built in feature extraction process. Second, we will evaluate the number of parents allowed during DAG structure searching using K2 algorithm. Third we will evaluate the initiation order when using K2 algorithm, the first initiation we will use nodes order that represent naive bayes classifier, and the second initiation we will use random order for the nodes. The evaluation measures that we will be using are accuracy, precision, recall, and F1-Measure. Our performance measures result is the average of n-fold cross-validation results with \( n = 20 \).

4.1 Features Evaluation

In this task we will evaluate features quality by creating several combinations of features. We will observe which feature combination returns the best result. The goal is to find out whether the feature is worth to keep during identifying paraphrase or not. As we know the more features means the more nodes exist in the DAG, which means adding time and space complexity. The features combination performance is explained in Table 2.

| No | Features                | Accuracy | Precision | Recall | F1-Score |
|----|-------------------------|----------|-----------|--------|----------|
| 1  | LEV + COS + LCS + WUP + PATH | 0.756    | 0.752     | 0.765  | 0.758    |
| 2  | LEV + COS + LCS + WUP     | 0.757    | 0.753     | 0.765  | 0.758    |
| 3  | LEV + COS + LCS + PATH    | 0.750    | 0.745     | 0.759  | 0.752    |
| 4  | LEV + COS + WUP           | 0.749    | 0.770     | 0.709  | 0.738    |
| 5  | COS + LCS + PATH          | 0.729    | 0.727     | 0.733  | 0.730    |
| 6  | COS + LCS + WUP           | 0.725    | 0.719     | 0.739  | 0.729    |
| 7  | COS + WUP + PATH          | 0.744    | 0.776     | 0.685  | 0.728    |
| 8  | LEV + WUP + LCS + PATH    | 0.720    | 0.713     | 0.753  | 0.724    |
| 9  | LEV + LCS + PATH          | 0.715    | 0.710     | 0.727  | 0.715    |
| 10 | LEV + LCS + PATH          | 0.712    | 0.705     | 0.722  | 0.713    |

Where:

LEV = Levenshtein distance

COS = Tf based cosine similarity

LCS = Longest common subsequence

WUP = Wu and Palmer

PATH = Shortest path
We can see that that the number of combination displayed in in the Table 2 is ten. The reason of that is we take combination number 10 as performance baseline. Other possible feature combinations which are not displayed in the Table 2 have lower score that the baseline. So, we believe ten combinations are sufficient to represent performance of the model.

Based on features performance results in Table 2 we conclude that the best combination is including all features in this research. The combination returns highest F1-Score measure with 75.8%, although the combination features without PATH feature gives us same F1-Score, higher precision and higher accuracy. We believe that this is not enough reason to remove the PATH feature from the model. Instead of removing the PATH feature to gain very little improvement on performance, we should try to improve or modify the feature.

We also believe that this performance result, especially for PATH feature caused by limited dataset which only has 1004 data. The size of dataset available leaves to choose n-fold cross-validation which means the evaluation performance is averaged through testing process on known environment. On PATH feature function we know that it searches the shortest path between pair of words, but turns out that the shortest path doesn’t necessarily represent similarity of the word. The PATH feature may be useful on data test that is properly separated from the development process.

Up until now we only talk about first two feature combination, if we look at feature number three and eight we can see that they have same number of features. The difference between them is the feature number three has COS feature, and the feature number eight has WUP feature. That only difference impacts performance quite significant, feature number three 3.4% better than feature number eight. We call it significant because from ten combination we have, the combination number eight is closer to the base line with only 1.1% higher than the baseline. It means difference between combination number eight and ten is one third of difference between combination number eight and three. Even if we compare the combination number eight with other combination that have COS feature, the performance of combination eight is less than others, even when others have less feature. It goes without saying that COS feature holds better overall performance compared to WUP feature.

We conclude that every feature is important to the model, but the significance is different. The most significant feature is COS feature, it’s absence on the model impacts the performance quite a lot. On the other hand, the least significant feature is PATH feature, it’s presence has yet gives us any significant impact to the performance. For other features we calculate the order of significance based on performance in Table 2. From highest significance to lower significance we have COS, LEV, WUP, LCS, and PATH. The order of significance is based on the amount of performance drop if we exclude the feature. For example in our case, the absence of COS feature is causing performance drop greater than WUP feature, it means the COS feature significance is higher than WUP feature. Even the significant between four features (COS, LEV, WUP and LCS) is different doesn’t mean one feature should be removed. Because the absence of one feature in the list will return lower performance score than combination of the four features.

4.2 Evaluation on number allowed parents on K2 Algorithm
The purpose of this evaluation is to find out the outcome of different u (max allowed parents in a node) value to the DAG structure and its impact to the performance. As we know that K2 algorithm uses a parameter that limits allowed parents in the DAG structure. In the observation we will use 1, 2, and 3 as u number. The scoring function used in the K2 algorithm is BDeu scoring function explained in equation 4. The DAG and the joint probability generated by K2 algorithm as follows.
The joint probability equation for DAG where $u = 1$ is:

$$P(CLASS, LEV, COS, LCS, WUP, PATH) = P(LEV|CLASS)P(COS|CLASS)P(LCS|CLASS)P(WUP|CLASS)P(PATH|CLASS)P(CLASS)$$  \hspace{1cm} (8)

The joint probability equation for DAG where $u = 2$ is:

$$P(CLASS, LEV, COS, LCS, WUP, PATH) = P(LEV|CLASS)P(COS|CLASS)P(LCS|CLASS, COS)P(WUP|CLASS, LCS)P(PATH|CLASS, WUP)P(CLASS)$$  \hspace{1cm} (9)

The joint probability equation for DAG where $u = 3$ is:

$$P(CLASS, LEV, COS, LCS, WUP, PATH) = P(LEV|CLASS)P(COS|CLASS)P(LCS|CLASS, COS)P(WUP|CLASS, LCS)P(PATH|CLASS, WUP)P(CLASS)$$  \hspace{1cm} (10)
After we obtain DAG structures, we attempt to generate two random DAG as comparison to the structure generated by K2 algorithm.

\[ P(\text{CLASS}, \text{LEV}, \text{COS}, \text{LCS}, \text{WUP}, \text{PATH}) = P(\text{LEV}|\text{CLASS}, \text{COS})P(\text{COS}|\text{CLASS}, \text{LCS})P(\text{LCS}|\text{CLASS})P(\text{WUP}|\text{PATH})P(\text{PATH}|\text{LCS, CLASS})P(\text{CLASS}) \]  

(11)

\[ P(\text{CLASS}, \text{LEV}, \text{COS}, \text{LCS}, \text{WUP}, \text{PATH}) = P(\text{CLASS}|\text{LEV})P(\text{COS}|\text{CLASS})P(\text{LCS}|\text{CLASS})P(\text{WUP}|\text{CLASS})P(\text{PATH}|\text{WUP})P(\text{LEV}) \]  

(12)

The BDeu score and average performance of all three DAG from K2 algorithm and two random generated DAG are explained in Table 3 as follows.

| No | DAG                          | BDeu    | Accuracy | Precision | Recall  | F1-Score |
|----|-----------------------------|---------|----------|-----------|---------|----------|
| 1  | DAG where u = 1             | -4764.234 | 0.716    | 0.727     | 0.691   | 0.708    |
| 2  | DAG where u = 2             | -4028.149 | 0.756    | 0.752     | 0.765   | 0.758    |
| 3  | DAG where u = 3             | -4028.149 | 0.756    | 0.752     | 0.765   | 0.758    |
| 4  | Generated DAG 1             | -4032.979 | 0.748    | 0.740     | 0.762   | 0.751    |
| 5  | Generated DAG 2             | -4316.510 | 0.729    | 0.740     | 0.705   | 0.722    |

On table 5 we find that the DAG structure generated by K2 algorithm for current features only able to generate two kinds of structure. The first structure is the same with naive bayes structure where the u value is 1. The reason of it because parameter only allows one parent of each node, it means the algorithm will find the best parent for each node. In our case the best parent to our nodes according to K2 algorithm is the class itself, that’s why the algorithm returns naive bayes structure.

The second DAG structure generated by K2 is on figure 3 and 4. We notice that there is no difference between DAG on figure 3 and 4. It means the K2 algorithm has found the best DAG it can find. If we
use u with value 5, 6, ..., n we will not get any differences. That’s because the number of max parent allowed depends on the nodes. Either the number of nodes or the CPT (Conditional Probability Table) of the nodes will affect DAG structure searching. Our total nodes in the graph is 6 which means maximum number of nodes that are assigned as parents is 5 nodes. But with all of that, the result instead gives us DAG in figure 3 and 4 which max parent in its DAG is two parents per node. The cause of it because as mentioned before the generation of the DAG structure depend on CPT in the nodes. The CPT will affect BDeu, which means BDeu score is better on max parents = 2 than max parents > 2.

We use the DAG number 1 in Table 3 as baseline because it takes form of naive bayes classifier, as we know in many cases naive bayes classifier can be used as baseline. From performance point of view we notice that the relation between BDeu score and performance is linear with each other. It means that the higher BDeu score will lead us to higher performance score. The DAG number 2 and 3 perform best, with F1-Score is 75.8%. They performance scores are higher than other two random generated DAG. Even two random generated DAGs have better score than the baseline.

4.3 Evaluation of nodes order in K2 Algorithm
In this evaluation we will observe the model performance when we change nodes order in K2 algorithm. The first case we will use order that represent naive bayes classifier and the second case is random order. For the random nodes order we will perform classification 10 times and take the average score between runs. The result of nodes order observation as follows in Table 4.

| No | DAG                                | BDeu   | Accuracy | Precision | Recall | F1-Score |
|----|------------------------------------|--------|----------|-----------|--------|----------|
| 1  | Fixed Initiation with naive bayes order | -4028.149 | 0.756    | 0.752     | 0.765  | 0.758    |
| 2  | Random order run 1                 | -4028.177 | 0.752    | 0.747     | 0.763  | 0.755    |
| 3  | Random order run 2                 | -4028.156 | 0.753    | 0.754     | 0.751  | 0.752    |
| 4  | Random order run 3                 | -4028.428 | 0.759    | 0.758     | 0.761  | 0.759    |
| 5  | Random order run 4                 | -4028.149 | 0.746    | 0.747     | 0.745  | 0.746    |
| 6  | Random order run 5                 | -4028.400 | 0.751    | 0.752     | 0.749  | 0.750    |
| 7  | Random order run 6                 | -4039.326 | 0.746    | 0.743     | 0.753  | 0.748    |
| 8  | Random order run 7                 | -4028.349 | 0.747    | 0.745     | 0.751  | 0.748    |
| 9  | Random order run 8                 | -4028.428 | 0.747    | 0.743     | 0.755  | 0.749    |
| 10 | Random order run 9                 | -4028.149 | 0.750    | 0.748     | 0.755  | 0.751    |
| 11 | Random order run 10                | -4039.299 | 0.755    | 0.758     | 0.749  | 0.754    |
| 12 | Average between run 1 – 10         | -4030.49  | 0.750    | 0.749     | 0.753  | 0.751    |

Based on results in table 4 we can see that the fixed nodes order gives us better performance result than random order. It’s because the not only the performance score but also the consistency of the performance. If we look at random order between run 1 to run 10 we can see constant changes in performance scores. Although there is a run that returns better result than fixed order in run three, it’s just slightly better. The chance for us to encounter such result is one in ten runs, it’s not justified to use random order in pursue of slightly better performance when the difference actually is ignorable. From average result’s point of view, we also find that the overall score of random order is noticeably lower than fixed order.
5 Conclusion and future work

Indonesian paraphrase identification is conducted by performing comparison of pair phrase/sentence whether they are paraphrase or not. During the process we perform preprocessing, feature extraction, classifier building and evaluation. During feature extraction we perform two types of feature, they are syntactic and semantic features. Syntactic features consist of normalized levenshtein distance (LEV), tf cosine similarity (COS), and longest common subsequence (LCS). For semantic feature we have Wu and Palmer feature (WUP) and shortest path feature (PATH). The mentioned features is used in Bayesian Networks classifier.

In this research we can conclude that by performing classification with our method on Indonesian Paraphrase Dataset with our method we obtain average accuracy 75.6%, precision 75.1%, recall 76.5% dan F1-Measure 75.8%. When performing this research, we find difficulties especially in the thesaurus area, where there is no Indonesian thesaurus that is ready to be used with our method. We are choosing alternative by translate the word tokens with Google Translator API, but the problem with that is the ambiguity of translation result. For example if we have a word in Indonesian “tanah” and “air” if we translate the word separately it will mean earth and water, but if we look at another context it can mean homeland. With our method we still encounter problem with hyponym, homonym, polysemy and homograph. We also encounter difficulties with complex sentence that consists of a clause and main clause. In Indonesian Language, the main clause is the main idea of an expression, and the clause is supporting the main clause. With this method we were not able to extract the main idea of complex sentence. Our guess is we need specific rules that able to specify main clause and support clause in a complex sentence to extract its main idea.

From performance point of view, we conclude that the five features (COS, LEV, WUP, LCS, and PATH) should remain in the classifier model. That’s because the combination has the highest performance scores along with combination of four features (COS, LEV, WUP, and LCS). We mention that the current PATH feature doesn’t have any significant result. It doesn’t mean we should remove the feature from the model, because there’s possibility that our data doesn’t fit quite well with this feature. If we try with another data, there’s a chance that this feature (PATH) is having significant role. Speak of feature significance, we conclude that COS feature is the most significant feature, followed by LEV, WUP and LCS. The significance is measured by how much performance drop occurred when a feature is excluded from the model.

For K2 algorithm we conclude that the best number of allowed parents per node are two and the ordering of initiation is better off with the fixed one with naive bayes order. The two conditions are based on performance result during evaluation.

As future work we may try to build a feature model that able to recognize complex sentence. We also have to create feature extraction method that able to extract feature about passive or active sentence, positive or negative sentence (polarity) and significance of each word in sentence/phrase. We also need to develop an feature extraction method that ready to be used if in the future Indonesian Thesaurus is fully completed.

References

[1] A. Reshamwala, D. Mishra and P. Pawar, "Review On Natural Language Processing," ACST – Engineering Science and Technology: An International Journal (ESTIJ), vol. 3, no. 1, 2013.

[2] KBBI, "KBBI - Parafrasa," [Online]. Available: kbbi.web.id/parafrasa.

[3] J. Asian, H. E. Williams and S. Tahaghoghi, "Stemming Indonesian," RMIT University, Melbourne.

[4] U. B. Kjaerulff and A. L. Madsen, Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, New York: Springer, 2012.
[5] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques, Cambridge, Massachusetts: The MIT Press, 2009.

[6] Heckerman and David, A Tutorial on Learning With Bayesian Networks, Redmond: Microsoft Corporation, 1996.

[7] A. Abdi, N. Idris, R. M. Aliguliyev and R. M. A., "PDLK: Plagiarism Detection Using Linguistic Knowledge," Expert Systems With Application, pp. 1-11, 2015.

[8] R. Barzilay and K. McKeown, "Extracting paraphrases from a parallel corpus," Annual Meeting of the Association for Computational Linguistics, pp. 50-57, 2001.

[9] Y. Shinyama, S. Sekine, K. Sudo and R. Grishman, "Automatic paraphrase acquisition from news articles," 2002.

[10] A. A. Pratiwi, "On The Feature Selection and Classification Based on Information Gain for Document Sentiment Analysis," Applied Computational Intelligence and Soft Computing, 2017.

[11] M. Mubarok, Adiwijaya and M. Aldhi, "Aspect-based sentiment analysis to review products using Naive Bayes," In AIP Conference Proceedings, vol. 1867, no. 1, 2017.

[12] N. P. A. Vo, S. Magnolini and O. Popescu, "Paraphrase Identification and Semantic Similarity in Twitter with Simple Features," Proceedings of Social NLP, pp. 10-19, 2015.

[13] Adiwijaya, M. S. Mubarok and A. H. Prayogo, "Identifikasi Parafrasa Pada Dokumen Teks Bahasa Indonesia Menggunakan Bayesian Networks," Indonesia Symposium On Computing, 2016.

[14] R. M.Alguliev, R. M.Aliguliyev and C. A.Mehdiyev, "Sentence selection for generic document summarization using an adaptive differential evolution algorithm," Swarm and Evolutionary Computation, vol. 1, no. 4, pp. 213-222, 2011.

[15] M. S. K, D. K. C. Shet and D. U. Acharya, "A New Similarity Measure For Taxonomy Based On Edge Counting," International Journal of Web & Semantic Technology (IJWesT), vol. 3, no. 4, 2012.

[16] M. Palmer and Z. Wu, "Verb Semantics and Lexical Selection," In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994.

[17] J. Ferlež and M. Gams, "Shortest-Path Semantic Distance Measure in WordNet v2.0," Informatica, vol. 28, pp. 381-386, (2004).

[18] V. Levenstein, "Binary codes capable of correcting deletions, insertions, and reversals," Soviet Physics Doklady 10, vol. 8, pp. 707-710, 1966.

[19] B. Bukh and J. Ma, "Longest common subsequence in sets of words".

[20] B. Malone, "Scoring Functions for Learning Bayesian Networks," University of Helsinki, Finland, 2014.

[21] G. F. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data," Machine Learning, no. 9, pp. 309-347, 1992.