Received: January 9, 2024. Revised: March 12, 2024.

INASS

*International Journal of*
Intelligent Engineering & Systems

http://www.inass.org/

# Performance Analysis of End-to-End Neural Coreference Resolution in English and Indonesian Texts

**Gunawan[1]*** **Ivan Hosea [1]** **Esther Irawati Setiawan[2]** **Kimiya Fujisawa[3]**

[1]*Informatics Department, Institut Sains dan Teknologi Terpadu Surabaya, Indonesia*
[2]*Information Technology Department, Institut Sains dan Teknologi Terpadu Surabaya, Indonesia*
[3]*School of Media Science, Tokyo University of Technology, Tokyo, Japan*
* Corresponding author's Email: gunawan@stts.edu

**Abstract:** This research evaluates the performance of End-to-End Neural Coreference Resolution models in English and Indonesian linguistic contexts, drawing particular attention to the model by K. Lee, recognized for its simplified preprocessing methodology. In this research, we use raw text to find and link mentions in documents, and handle different languages. For the evaluation metrics, the English model achieved F1-Scores of 67.94% and 67.14% on the OntoNotes-5.0 development and training sets. The Indonesian model, prepared using a CoNLL-2012 formatted dataset, attained an F1-Score of 68.88% on a 25% segment of the Book of Markus. We also analyzed the additional features integrated into the model to assess their contributions to performance improvement. The findings indicate that while the English model demonstrates generalizability across various coreference challenges, the Indonesian model's performance is more domain-specific, being particularly effective within the confines of the Book of Mark.

**Keywords:** Coreference resolution, End-to-end learning, Deep learning, Natural language processing.

## 1. Introduction

Machine learning is an algorithm used by computers to learn certain tasks that cannot be applied with conventional programming [1]. The easiest example of the task in question is work that can be easily done by humans, such as recognizing language, seeing, hearing, and so on. The machine learning task in the field of human language recognition is Natural Language Processing [2]. One of the tasks of Natural Language Processing is Coreference Resolution.

Coreference Resolution is a process of determining if 2 or more expressions in natural language are the same entity in the real world [3]. Coreference Resolution is also applied to assist other natural language processing tasks, namely Question Answering [4], Sentiment Analysis [5], Document Summarization, Quote Attribution, and Information Extraction.

Research on Coreference Resolution has been carried out in many foreign languages, but not in Indonesian. Previous coreference studies that have the best results [6, 7], use Syntactic Parser and Mention Detector algorithms made by themselves. The problem that arises is if there is a parsing error it can cause cascading errors. In addition, the custom Mention Detector algorithm tends to be unusable in other languages. End-to-End Neural Coreference is a method that does not use a Mention Detector or Syntactic Parser, but has better performance than the previous methods.

In recent times, the exploration of End-to-End Neural Coreference Resolution (E2E NCR) has garnered attention due to its capability to address previous methodological shortcomings. Particularly, this research delves into the applicability of E2E NCR models within Indonesian linguistic contexts, marking a significant stride towards understanding coreference resolution in under-studied languages.

By leveraging a simplified preprocessing methodology, as proposed by K. Lee, this work presents an insightful evaluation, shedding light on the potential and limitations of E2E NCR models in handling coreference challenges in Indonesian texts, hence contributing to the broader discourse on advancing coreference resolution techniques in multilingual[8] settings.

This research also utilizes a method that combines pre-processing and avoids cascading errors by bypassing the need for syntax parsers or mention detectors. It uses additional features such as speaker and genre information to improve coreference resolution performance across diverse linguistic domains and contexts. This method also achieves competitive results on the English OntoNotes benchmark and outperforms existing methods on the Indonesian Book of Mark dataset.

Moreover, the robust analysis carried out, encompassing a comparative study between English and Indonesian models, not only underscores the domain-specific efficacy of the Indonesian model but also propels further inquiry into optimizing E2E NCR for diverse linguistic landscapes. The meticulous examination of additional features integrated into the model, and their consequent impact on performance enhancement, embodies a novel endeavour towards refining coreference resolution systems. This investigation, therefore, not only furnishes a valuable benchmark for evaluating E2E NCR models in Indonesian but also catalyses future explorations aimed at transcending linguistic boundaries in coreference resolution tasks.

The following section discusses the previous studies on Coreference Resolution. Next, the proposed method of End-to-End Neural Coreference Resolution is explained in Section 3. Section 4 introduces the Image-Text dataset for Indonesian Sentiment Analysis. The experimental results and analysis are presented in Section 5. The last section summarizes the conclusion and future directions.

## 2. Literature review

Coreference Resolution is one of the tasks in Information Extraction and Natural Language Processing. The term ‘Coreference Resolution’ itself is derived from the linguistic field, referring to the concept of coreference. In the following section, we will explore the fundamentals of Coreference Resolution, including Natural Language Processing and Named Entity Recognition [9]. Subsequently, we will provide an explanation of Coreference Resolution itself.

Natural Language Processing is a technique that involves the computational representation and analysis of human language. This is achieved through various methods such as tokenization, parsing, tagging, and semantic analysis. However, this technique is not without its drawbacks. For instance, the ambiguity, variability, and complexity of natural language can affect it, making it difficult to capture the meaning and context of texts. Additionally, it can be dependent on the availability and quality of linguistic resources like corpora, lexicons, grammars, and ontologies, which can vary across languages and domains. Furthermore, the emergence of new words, phrases, and genres can pose a challenge, necessitating constant updates and adaptations of the models and methods [10].

Named Entity Recognition is a subtask of Information Extraction. Named Entity Recognition aims to find entities in text documents and classify these entities into various categories such as person’s name, organization’s name, time, and so on. Named Entity Recognition is an important piece of research in Natural Language Processing because it can be used to help with more complicated tasks such as Question Answering, Machine Translation, and Coreference Resolution.

Coreference resolution[9] has been explored using various methods as displayed on Table 1, from traditional machine learning to advanced neural networks. While neural models often perform well, they require considerable computational resources and may lack interpretability. The importance of adaptable natural language processing techniques and named entity recognition is highlighted to address the evolving complexities of language and improve coreference resolution systems.

Taking a fresh perspective on end-to-end neural coreference resolution, a new baseline model is introduced in this study. Despite its simplicity, it surpasses more complex recent models on the public English OntoNotes benchmark. This underlines the importance of carefully considering the complexity of both existing and new models, as even minor modifications can yield improved results. [11]

Meanwhile, another study [12] delves into enhancing end-to-end neural coreference resolution. It presents a novel algorithm to adjust coreference clusters, capable of eliminating mentions that lack similarity and reducing errors caused by inconsistent coreference clusters throughout a document. The model’s performance is further enhanced by replacing the simple scoring function used to compute head word scores for the attention mechanism with a feed-forward neural network, and by modifying the maximum length of a mention. These alterations significantly boost the performance of coreference resolution. Subsequently, a new system for resolving coreference in Korean text is introduced in [13]. The system employs a neural network with an attention mechanism to identify and

Table 1. Previous Approaches on Coreference Resolution

| Author(s) | Techniques Used | Objective | Merits | Disadvantages |
|---|---|---|---|---|
| W. M. Soon, H. T. Ng, and D. C. Y. Lim[3] | Machine Learning | Coreference resolution of noun phrases | Not mentioned | Lack of clarity on the specific approach and potential limitations |
| K. Clark and C. D. Manning[6] | Neural Network | Incorporation of entity-level information | Outperforms the current state-of-the-art | Neural networks may require substantial computational resources and extensive training data. Interpretability can be challenging. |
| S. Wiseman, A. M. Rush, and S. M. Shieber[7] | Recurrent Neural Networks | Learning global features for coreference resolution | Outperforms the current state-of-the-art | RNNs may suffer from vanishing gradient problems and can be computationally expensive during training. |
| S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang[8] | Comparison of various conventional machine learning method such as Logistic Regression, MaxEnt, C4.5, and others. | Modeling multilingual unrestricted coreference in OntoNotes | Not mentioned | Lack of details on the specific models used and their limitations. |
| K. Lee, L. He, M. Lewis, and L. Zettlemoyer[19] | Neural mention-ranking model | End-to-end neural coreference resolution | Outperforms all previous work | Neural models may be sensitive to hyperparameters and require extensive tuning. Training large neural models can be resource-intensive. |
| K. Clark and C. D. Manning[20] | Deep Reinforcement Learning | Mention-Ranking Coreference Models | Outperforms the current state-of-the-art | Reinforcement learning models can be challenging to train, and their performance heavily depends on reward design and exploration strategies. |

link all mentions of the same entity in a document. Initially, all nouns in the document are considered as potential mentions. The attention mechanism then learns to predict the position of the referenced entity for each noun. Experimental results demonstrate that this system outperforms all other known systems on Korean language coreference resolution.

In a similar vein, a system for resolving coreference in Indonesian text is presented [14]. This system utilizes a deep neural network to learn how to identify pairs of words that refer to the same entity, and a singleton classifier to prevent words that only refer to themselves from being grouped together. The system surpasses the best existing system, achieving a high score on a standard benchmark for coreference resolution.

A novel approach to coreference resolution was proposed in [15]. Traditional models heavily rely on span representations to find coreference links between word spans, which requires various pruning techniques due to computational complexity. It was suggested to consider coreference links between individual words instead of word spans and then reconstructing the word spans. This method reduces the complexity of the coreference model and allows it to consider all potential mentions without pruning any of them out. The paper also demonstrates that with these changes, SpanBERT for coreference resolution will be significantly outperformed by RoBERTa. Despite its efficiency, the model performs competitively with recent coreference resolution systems on the OntoNotes benchmark.

Coreference resolution is also implemented in Indonesian as in [16], which explores the application of a multi-pass sieve coreference resolution model to the Indonesian language. The multi-pass sieve is a deterministic coreference model that implements several layers of sieves, each taking a pair of correlated mentions from a collection of non-coherent mentions. This model operates on the principle of high precision, followed by increased recall in each sieve. The authors conducted an experiment on 201 Wikipedia documents, and the multi-pass sieve system yielded a 72.74% MUC F-measure and a 52.18% BCUBED F-measure. This research demonstrates the potential of the multi-pass sieve technique for Indonesian coreference resolution tasks.

In a recent study, researchers explored the use of lexical and shallow syntactic features for resolving coreference in Indonesian text [17]. Their system successfully addressed coreference between pronouns and proper nouns, as well as between proper nouns and pronouns. To achieve this, they leveraged various features, such as identifying apposition relationships, determining the nearest candidate for a pronoun, analyzing sentence context, and examining preceding and following words. Remarkably, their system achieved a commendable score of 71.6% on a standard benchmark for coreference resolution [18].

## 3. Methodology

End-to-End Neural Coreference Resolution is a Coreference Resolution [21, 22] model that uses machine learning in its application. This model is considered end-to-end because the process that is executed when changing input to output is only 1 architectural model.

In Fig. 1, the first part input. The input to this network is a combination of word embedding and character embedding. The word embedding used is GloVe’s pre-trained word embedding with 300 dimensions and Turian’s word embedding with 50 dimensions.

Next, Bidirectional LSTM is used to encode lexical information from inside and outside each span. In the End-to-End Neural Coreference Resolution architecture, the input vector will be feed-forward to the Bidirectional LSTM. LSTM, which stands for

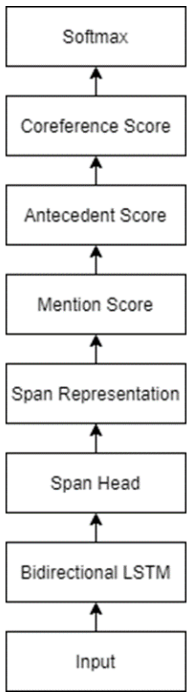


Figure. 1 Block Diagram End-to-End Neural Core Resolution Architecture

Long Short-Term Memory, is a development of the Recurrent Neural Network model, which is a model commonly used for processing sequential data such as text documents because the previous data in the sequence also determines predictions for subsequent data. Different from Recurrent Neural Network, LSTM has memory and also a gating system to overcome the vanishing gradient problem. Because LSTM only makes predictions to the data after it, Bidirectional LSTM was created which can make predictions to the data before and after it. In the following subchapters we will explain the vanishing and exploding gradient problem, weight initialization, and the Bidirectional LSTM architecture used.

After that, the span head will be formed as a feature primarily to help represent spans whose width is more than 1. Then the Bidirectional LSTM and span head features will be some of the features used to form the span representation. After getting the span representation, the Mention Score will be calculated.

Mention Score is a value of how likely a span is a mention or entity. Only spans with the highest value will have their Antecedent Score assessed, the rest will be prune. Antecedent Score itself is a value of how likely it is that a span is an antecedent of another span.

After getting the Antecedent Score, Mention Score and Antecedent Score will be used to calculate the Coreference Score. After the Coreference Score is obtained, softmax will be carried out to determine the antecedents of each span selected as a mention.

### 3.1 Task coreference resolution

In the End-to-End Neural Coreference Resolution model, the input required from the dataset includes the main document and speaker and genre metadata features. The speaker metadata feature indicates who the speaker of the word part of the document is, while the genre feature indicates the genre of the document. The output of the following model is a cluster of mentions that are considered to have a coreference relationship with each other.

The task is formulated with an input document *D* which has the number of words *T*. All words in the document will be represented in vector form containing Word Embedding, Character Embedding, and other features. Apart from words, other tokens such as punctuation marks will also be included in the input sequence. Of all the tokens in the document, the maximum number of spans that will be formed can be formulated using the following formula:

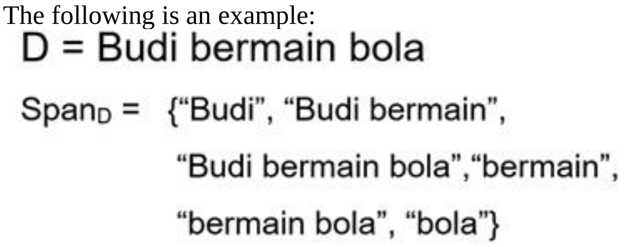
$$N = (T(T+1))/2 \tag{1}$$

In the formula above, N is the number of spans. For example, if there are 3 words in a document, then based on the formula above you will get 6 types of span.

In Fig. 2, the span will be formed from pieces of 1 word, 2 words, and 3 words. If there are T words, then the largest span is formed from T words. However, because 1 document can have thousands of words, forming a span throughout the document can take a very long time, and existing entities tend not to be very long, so forming the length of the span will be limited to the desired word length. If in the example document "*Budi bermain bola*" (Budi plays football) is limited by forming a span length of 1, then the spans that will be formed are only "*Budi*", "*bermain*" (plays), and "*bola*" (football). Some spans will then be selected as mentions, the rest will not be searched for antecedents.

The task aims to fill in the antecedents of the selected spans as mentions, for each mention *i* has an antecedent $y_i$. The set or possible candidates for each $y_i$ are as follows:

The following is an example:


D = Budi bermain bola

Span$_D$ = {"Budi", "Budi bermain", "Budi bermain bola","bermain", "bermain bola", "bola"}

Figure. 2 Example of a Span Formed

Table 2. Examples of Candidate Antecedents

| Mention Number | Mention Name | Candidate Antecedent *Y(i)* |
|---|---|---|
| 1 | Budi | {ϵ} |
| 2 | *Lapangan* (field) | {ϵ,1} |
| 3 | *pemilik lapangan* (field owner) | {ϵ,1,2} |
| 4 | *Dia* (he) | {ϵ,1,2,3} |
| 5 | *Setelah* (after) | {ϵ,1,2,3,4} |
| 6 | *anak itu* (that child) | {ϵ,1,2,3,4,5} |
| 7 | *tempat itu* (that place) | {ϵ,1,2,3,4,5,6} |

$$y_i = \{\epsilon, 1, 2, \ldots, i-1\} \tag{2}$$

From the set above, $\epsilon$ is epsilon. The epsilon antecedent dummy indicates that mention *i* has no antecedent. The number 1 in the set above indicates mention 1, as well as mention 2. The candidate antecedent $y_i$ only reaches *i*-1 because it will only check with previous mentions. For example, there is a small document "*Budi sedang bermain bola di lapangan ketika pemilik lapangan menyuruh dia pulang. Setelah itu pulanglah anak itu meninggalkan tempat itu*" (Budi was playing football on the field when the field owner told him to go home. After that the child went home and left the place), and for example the span chosen as a mention is "*Budi*", " *lapangan*" (field), "*pemilik lapangan*" (field owner), "*dia*" (he), "Setelah" (after), "*anak itu*" (that child), and "*tempat itu*" (that place), then the candidate mentions from each mention can be described in Table 2.

In Table 2, the mention "*Budi*" only has a candidate antecedent ϵ because it is the first mention. The mention "*lapangan*" (field) has a candidate antecedent ϵ and 1 which is the mention of "*Budi*" because the mention of "*Budi*" comes before the mention of "*lapangan*" (field). Likewise, other mentions both have a candidate antecedent ϵ and all mentions that come before that mention.

After finding all the antecedents of each mention, clusters will be formed containing mentions that have a coreference relationship. The following is an example of a table after the antecedent is determined: In Table 3, it can be seen that the antecedent of "*Dia*" (he) is mention 1, namely "*Budi*", the antecedent of "*anak itu*" (that child) is "*Dia*" (he), and the antecedent of "*tempat itu*" (that place) is"lapangan" (field). If it is made into a cluster, 2 clusters will be created as in the Table 4.

As can be seen in Table 4, all mentions that have

Table 3. Examples of Antecedents

| Mention Number | Mention Name | Antecedent |
|---|---|---|
| 1 | Budi | $\epsilon$ |
| 2 | *Lapangan* (field) | $\epsilon$ |
| 3 | *pemilik lapangan* (field owner) | $\epsilon$ |
| 4 | *Dia* (he) | 1 |
| 5 | *Setelah* (after) | $\epsilon$ |
| 6 | *anak itu* (that child) | 4 |
| 7 | *tempat itu* (that place) | 2 |

Table 4. Cluster Example

| Cluster | Mention List |
|---|---|
| 1 | {“*Budi*”, “*Dia (he)*”, “*anak itu (that child)*”} |
| 2 | {“*lapangan (field)*”, “*tempat itu (that place)*”} |

only one antecedent relationship with one of the mentions in the cluster will be grouped into one cluster. Like in cluster 1 where the antecedent of “*Dia*” (he) is “Budi” and the antecedent of “*anak itu*” (that child) is “*Dia*” (he), but the three mentions are made into 1 cluster because there is at least 1 related coreference between the mentions.

This End-to-End model in Fig. 3 [19] starts from Word and Character Embedding. The Word Embedding used is GloVe [23] 6B 300d and Turian [24] 50d.

## 3.2 Word embedding

Word Embedding is a representation of words in the form of number vectors which aims to improve the performance of Natural Language Processing tasks. The vector of these words will become the architectural input.

Fig. 4 is an illustration of Word Embedding which has 7 dimensions. Each word is represented in vector form. It is illustrated that each dimension is a category that exists in the real world, such as how much value words have, namely living objects, humans, verbs, and so on with a value limit of -1 to 1. However, Word Embedding will not have labeled categories as above, but This illustration indicates that the larger the dimensions of the Word Embedding, the larger the categories that can be loaded. The closeness between vectors can be measured by the Eucledian distance. The Eucledian distance of 2 words represented in the Word

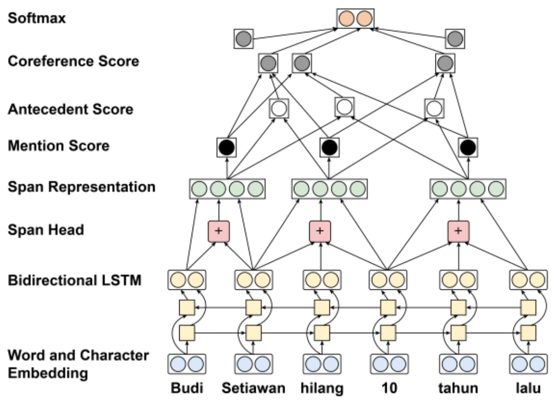


Figure. 3 End-to-End Neural Coreference Resolution Architecture

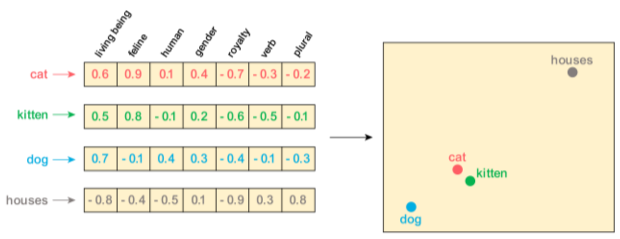


Figure. 4 End-to-End Neural Coreference Resolution Architecture

Embedding vector will be lower if the two words have a strong relationship. On the right side of the image, you can see that the words “cat” and “kitten” are close, indicating that the two words have a strong relationship.

In the End-to-End Neural Coreference Resolution architecture, there are 2 types of Word Embedding used, namely Word Embedding which is based on GloVe and also Word Embedding which is based on Turian. The two Word Embeddings are trained first before the End-to-End model is run. The End-to-End model only takes the contents of the finished results of the two Word Embeddings. In the subsequent subchapters, the two Word Embeddings will be explained.

## 3.3 GloVe

Global Vectors (GloVe) is a word representation model created by J. Pennington, R. Socher, and C. D. Manning in 2014. It is called Global Vectors because it uses global statistics from documents in the form of a word-word co-occurrence matrix. GloVe has been applied to the Word Analogy task and produces an accuracy of 75%16. The source code of GloVe is available open-source in C language and in Python which can be accessed in the Gensim library. Table 5 is an example of a word-word co-occurrence matrix

Received: January 9, 2024. Revised: March 12, 2024.

Table 5. Example of Co-occurrence Matrix

| | ***Budi*** | ***suka*** | ***membaca*** | ***menulis*** | ***cerpen*** |
|---|---|---|---|---|---|
| ***budi*** | 0 | 2 | 0 | 0 | 0 |
| ***suka*** | 2 | 0 | 1 | 1 | 0 |
| ***membaca*** | 0 | 1 | 0 | 0 | 0 |
| ***menulis*** | 0 | 1 | 0 | 0 | 1 |
| ***cerpen*** | 0 | 0 | 0 | 1 | 0 |
| **.** | 0 | 0 | 1 | 0 | 1 |

Doc 1: *Budi suka membaca*. (Budi likes reading)
Doc 2: *Budi suka menulis cerpen*. (Budi likes writing short stories)

Table 6. Examples of Probability and Ratios

| Probability and Ratios | *k* = *padat* | *k* = *gas* | *k* = *air* | *k* = *busana* |
|---|---|---|---|---|
| *P(k\|es)* | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| *P(k\|uap)* | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| *P(k\|es) / P(k\|uap)* | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

for 2 documents with a window size of 1. The window size determines the number of words before and after that will be counted in the co-occurrence matrix. For example, if the window size is 1, the word "*menulis*" (writing) in document 2 will count the words "*suka*" (like) and "*cerpen*" (short story) in the co-occurrence matrix, whereas if the window size is 2, the word "*menulis*" (writing) will count the word "*budi*", "*suka*" (like), "*cerpen*" (short story), and "." in the co-occurrence matrix.

The co-occurrence matrix will be denoted by X which has the elements $X_{ij}$, with *i* and *j* in $X_{ij}$ being unique words found in all documents. For example, *i* is *Budi* and *j* is like, then based on Table 5 $X_{ij}$ is 2. Then there is $X_i$ which is the number of times any word appears in context *i*. For example, if *i* is "*suka*" (like), then the value of $X_i$ is 4, because "*suka*" (like) appears twice in the context of "*Budi*", once in the context of "*membaca*" (reading), and once in the context of "*menulis*" (writing).

$$P_i = \frac{X_{ij}}{X_i} \tag{3}$$

The formula above itself is a probability formula for word *j* appearing in the context of word *i*. With this formula, the relationship between words can be determined as displayed in Table 6.

In Table 6, it can be seen that *k* is an entity in the real world, namely "*padat*" (solid), "*gas*" (gas), "*cair*" (liquid), and "*busana*" (fashion). If the "*es*" (ice) and "uap" (steam) probability values *k* are increasingly similar, then the ratio of the probability values between "*es*" (ice) and "*uap*" (steam) will be closer to 1, as can be seen look at *k* "*air*" (water) because both are water, and *k* "busana" (fashion) because both are not fashion.

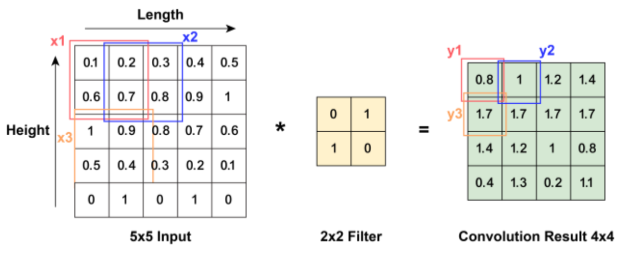


Figure. 5 Convolution Layer Architecture for Images

**Algorithm 1 Algorithm Turian**

```
E = Init_Word_Embedding()
N = Create_Ngram_List()
FOREACH X IN N
        Y = Corrupt(X)
   Xi = Map(X, E)
   Yi = Map(Y, E)
        Sx = FFNN(X)
        Sy = FFNN(Y)
        L = max(0, 1 - Sx + Sy)
        Learn()
NEXT X
```

Algorithm 1 is a Turian learning algorithm for 1 epoch. The Corrupt function is a function to change one of the words in n-gram X into another word. Map is a function for embedding each word in n-gram X stored in E. The FFNN function is useful for carrying out Neural Network[25], [26] operations from input, hidden, to output and returns the final score. L is the loss gained. The Learn procedure is used to carry out learning with backpropagation.

## 3.4 Character embedding

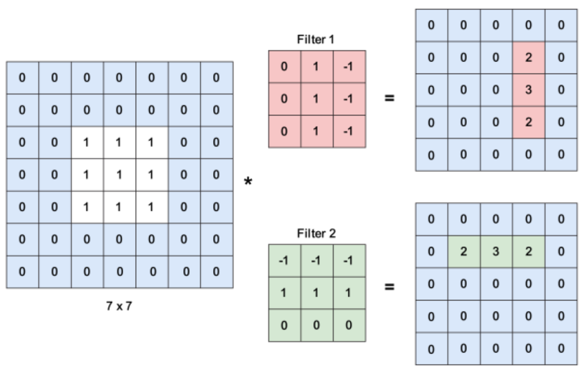


Figure. 6 Example of Pattern Detection with Convolution Layer

Received: January 9, 2024. Revised: March 12, 2024.

Table 7. Example of Mention Selection

| | |
|---|---|
| Setiawan | 1.7 |
| Setiawan bermain | -1 |
| Setiawan bermain bola | 2.5 |
| Bermain | -0.2 |
| Bermain bola | -1 |
| bola | 2.4 |

Sort
→

| | Span | M. Score |
|---|---|---|
| 🟩 | Budi Setiawan | 3.5 |
| 🟩 | Budi Setiawan | 2.7 |
| 🟨 | Setiawan bermain bola | 2.5 |
| 🟩 | Bola | 2.4 |
| 🟥 | Setiawan | 1.7 |
| 🟥 | Bermain | -0.2 |
| 🟥 | Setiawan bermain | -1 |
| 🟥 | Bermain bola | -1 |
| 🟥 | Budi Setiawan bermain | -2.3 |

| | |
|---|---|
| 🟩 | Selected as a mention |
| 🟥 | Pruned because Mention Score was too low |
| 🟨 | Pruned due to colliding with another span |

Character Embedding utilizes the characters in each word as features. Character Embedding uses a 1-dimensional Convolution Layer with ReLU activation followed by Max Pooling. Convolutional Neural Networks have been widely used to process images because they can detect patterns from small parts of the image.

In Fig. 5, the input is an image with a value that is closer to 0, the blacker it is and the closer it is to 1, the whiter it is and with a size of 5x5 pixels. What is done in the Convolution Layer is element-wise multiplication between the part of the input currently in the window and the filter. The window size is the same as the size of the filter, so in the image above the window size for a 2x2 filter is 2x2. Initially, the window will be in the top left corner, namely x1 in the image above. After doing element-wise multiplication with filters, exemplified in the image above with x1 ○ f where f is a 2x2 filter having the result placed in y1, then the window will then shift to the right by a stride. For example, stride is 1, then the window will shift by 1 pixel so that the window can be modeled with an x2 matrix. Then it will be multiplied again so that the result of x2 ○ f is placed in y2. The same process will be repeated continuously until the window cannot be shifted to the right. If the window cannot be shifted to the right, the window will return to the first pixel column and shift one pixel down. So, the window will be in position x3 according to the image above. Then the result x3 ○ f is placed in y3. After that, it will continue shifting to the right as usual. The multiplication process will complete when the window can no longer be shifted down.

Fig. 6 is an example of a filter as intelligence from the Convolution Layer which can detect patterns. The input image is a square image. There are 2 filters and the convolution results from filter 1 produce a square right line, while the convolution from filter 2 produces a square top line.

## 3.5 Mention score

Mention Score is a value of how big a span can have an antecedent. Spans that usually have an antecedent are spans which are nouns or pronouns. The input of the Mention Score is a Span Representation. Each Span Representation will have its own Mention Score. In calculating the Mention Score, Span Representation is fed forward to a Fully Connected Neural Network with a hidden layer. The output of the Neural Network is a scalar which becomes the Mention Score.

Because a document can have many spans, and only a very few of these spans can have coreference, after the Mention Score for each timestep is found, the collection of Mention Scores will be sorted and several spans with the highest Mention Score will be selected. If there are spans whose indexes collide, then only the span with the higher value is selected as a mention. The selected spans will then be searched for their Antecedent Score while the spans that are not selected will be pruned.

In Table 7, for example, the maximum span length is 2 and the number of spans selected as mentions is 75% of the number of words in the document, so the number of spans selected is 3. It can be seen that the span "*Setiawan bermain bola*" (Setiawan plays football) is pruned because it collides with the span "*Budi Setiawan*" whose Mention Score is higher. Collision because they both have the word "*Setiawan*". However, the "*Budi*" span is not prune even though the "*Budi Setiawan*" span also has the word "*Budi*" because the span is entirely contained within another span. So a span will not be prune if all of its contents are contained in another selected span or vice versa if it has the contents of another selected

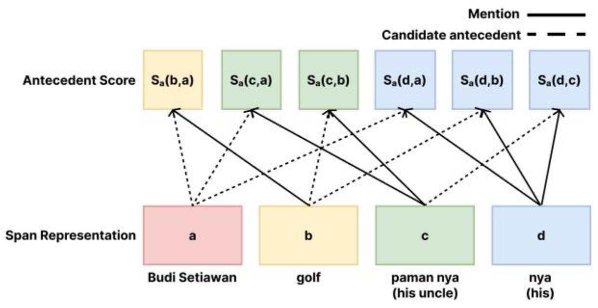


Figure. 7 Mention and Candidate Antecedents

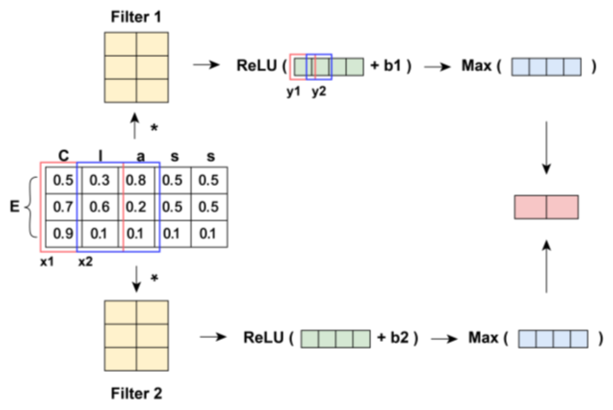


Figure. 8 End-to-End Neural Coreference Resolution Architecture

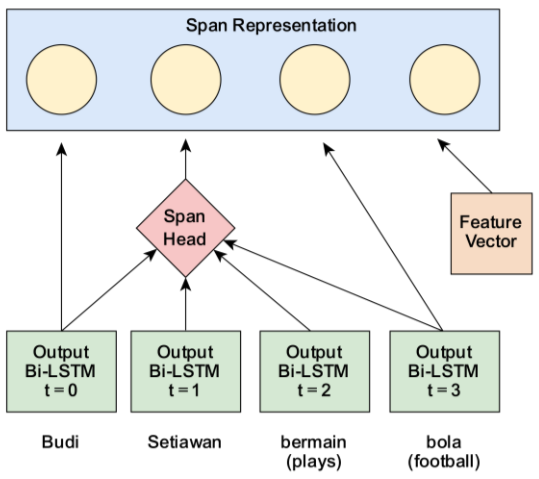


Figure. 9 Span Representation

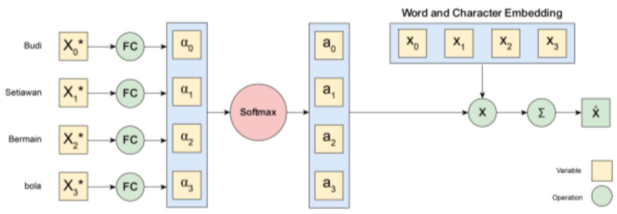


Figure. 10 Head Span Computation Graph

span.

## 3.6 Antecedent score

Antecedent Score is a value of how much a mention is an antecedent of another mention. Each mention has an Antecedent Score equal to the number of candidate antecedents minus 1. As explained in subsection 3.1, the candidate antecedents of a mention are other mentions that exist previously in the sequence and epsilon ϵ. Deducted by 1 because candidate ϵ does not have Span Representation.

Fig. 7 is an example of an Antecedent Score created from 4 mentions. In the image, an example document is "*Budi Setiawan bermain golf bersama paman nya*" (Budi Setiawan plays golf with his uncle) and the spans selected as mentions are "*Budi Setiawan*", "golf", "*paman*" (uncle), and "*nya*" (his). Antecedent Score is written in $S_a(i,j)$ where $i$ is a mention and $j$ is a candidate antecedent of mention $i$.

In Fig. 8, E is the embedding vector representation form of the character and "Class" is the word that will be converted into Character Embedding form. The first thing to do is map each character in the word into a vector form. For example, in the image above, the embedding of the character "s" is the vector [0.5, 0.5, 0.1]. In the image above, the stride value is 1 and the window length is 2x3, where 2 is the number of characters in 1 convolution and 3 is the dimension of the character features. These dimensions in the input image can be exampled with RGB coloring which requires 3 dimensions. Just like in the picture, the window will be multiplied by element-wise multiplication with the filter and then continued by moving the window to the right according to the stride size. After all the element-wise multiplication processes are complete, the results will be added by the respective bias filters, denoted by b1 and b2 in the image above, then this will be followed by ReLU activation to remove negative values. In the end, from the ReLU results of each filter, the element with the largest value will be selected and will be combined with the largest element from the other ReLU filters. The results of this concatenation will become input for the main architecture along with Word Embedding. Character Embedding is trained simultaneously with the main architecture.

In Fig. 9, the Span Representation is formed from the initial and final Bi-LSTM outputs of the span, the Span Head, and a Feature Vector as displayed in Fig. 4. The Feature Vector used is the embedding length of the span.

Mention Score[20] ($s_m$) is the value of how big a span is a mention. This value is calculated using an Artificial Neural Network[27], [28]. Only the few spans that have the highest value will be taken as a mention, the rest will be prune.

Received: January 9, 2024. Revised: March 12, 2024.

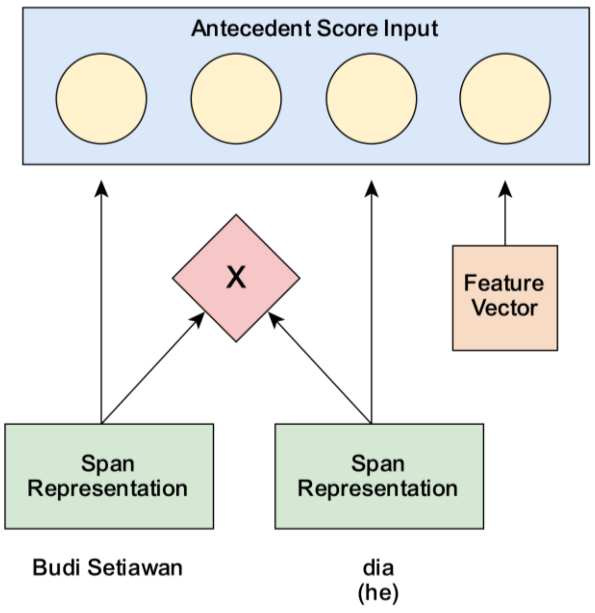


Figure. 11 Input Antecedent Score

**Algorithm 2 Algorithm Span Head**

```
01:   For I = 0 to len(Xbi) - 1
02:     Alpha[I] = FC(I)
03:   Next I
04:   A = Softmax(Alpha)
05:   Sh = Sum(a * X)
```

Algorithm 2 is the Span Head calculation, where X is the initial Word and Character Embedding input, Xbi is the Bidirectional LSTM output result, FC is the Fully Connected Layer function, and Sh is the final Span Head vector result.

Fig. 10 depicts an example of a computation graph from Span Head "Budi Setiawan bermain bola" (Budi Setiawan playing football) from input to output. In the graph, *xt** is the Bidirectional LSTM output. All Fully Connected Layers used in the graph above use the same weight. The output from the Fully Connected Layer in the graph above is the Head Score αt which is a scalar. All Head Scores are then collected into one vector and softmax is used for each element to get vector a.

The final Span Head is an embedding representation of the span. A Fully Connected Layer and Softmax are used to determine which word is more important to represent the span.

Fig. 11 is an example of input from an Antecedent Score where "*Budi Setiawan*" is a candidate antecedent and "*dia*" (he) is a mention. The input of an Antecedent Score is formed from the concatenation of the Span Representation of antecedent candidates, element-wise multiplication between the Span Representation of antecedent candidates and mentions, the Span Representation of mentions, and the Feature Vector.

Feature Vector input Antecedent Score consists of 2 features, namely distance and metadata. The distance feature is based on the distance between a mention and its antecedent candidate. For example, there is a document "Budi bermain golf bersama paman nya" (Budi plays golf with his uncle) and those selected as mentions are "*Budi*", "golf", and "*paman*" (uncle), then the distance between "*Budi*" and "*paman*" (uncle) is 2 because it is calculated from the difference in the mention index. The distance feature is represented by a random vector which will be trained simultaneously with the End-to-End model. The distance feature is separated into [0, 1, 2, 3, 4, 5-7, 8-15, 16-31, 32-63, 64+], where the number inside is the mention distance. For example, a candidate antecedent with a distance of 2 will have the same distance vector as another candidate antecedent with a distance of 2, while a candidate antecedent with a distance of 33 will have the same distance vector as a candidate antecedent with a distance of 57, because they are both found in range 32 to 63.

The metadata feature is a feature that takes information directly from the dataset. The metadata feature consists of 2, namely speaker and genre. Both features are represented by random vectors that are trained simultaneously with the End-to-End model. For the speaker feature, 2 random vectors are formed with one representing if the speaker of the mention and the candidate antecedent are the same, while the other represents if the speaker of the mention and the candidate antecedent are different. Genre features are divided according to the number of genres in the entire dataset and each different genre has a different embedding vector.

After the input from an Antecedent Score is formed, the input will be fed forward to the Fully Connected Neural Network which has a hidden layer. The output of the Neural Network is an Antecedent Score in scalar form.

Antecedent Score ($S_a$) has inputs like Fig. 6, namely Span Representation mentions, Span Representation candidate antecedents, the results of both element-wise multiplication, and Feature Vector. The following Feature Vector is speaker and genre metadata features as well as mention distance features, where each group of different mention distances has a different embedding, grouped by distance [0, 1, 2, 3, 4, 5-7, 8-15, 16- 31, 32-63, 64+]. Speaker metadata has 2 possibilities, namely the same or different speakers.

$$s(i,j) = \{0\ if\ \ j = \epsilon\ s_m(i) + s_m(j) + s_a(i,j)\quad if\ \ j \neq \epsilon \tag{4}$$

Received: January 9, 2024. Revised: March 12, 2024.

Above is the formula for Coreference Score *s(i,j)*. After obtaining the Coreference Score, the loss will be sought with the following marginal log-likelihood:

$$l = log \prod_{i=1}^{N} \sum_{y \in Y(i) \cap GOLD(i)} F(y) \tag{5}$$

$$F(y) = \frac{expexp\ (s(i,y))}{\sum_{y' \in y_i} exp(s(i,y'))} \tag{6}$$

Code End-to-End Neural Coreference Resolution developed by K. Lee accessible at https://github.com/kentonl/e2e-coref. The total LOC of all code is 1855. The code is written in Python and uses TensorFlow version 1.0.0 to apply the model architecture.

## 3.7 Learning

After the forward pass is carried out on the End-to-End Neural Coreference Resolution model, of course a backward pass will be carried out to study the task. This subchapter explains the backward pass that is carried out and explains several other things that are used when conducting training that are not discussed in the architecture.

## 3.8 Backpropagation

Backpropagation is the name of the technique used to perform a backward pass to find the gradient of each weight or bias that can be updated or studied. The gradient of a weight is the derivative of the loss function for the weight, so it can be written as follows.

$$Gr\ w = \frac{\partial L}{\partial w} \tag{7}$$

In the notation above, L is the loss and w is the weight whose gradient is sought.

## 3.9 Hyperparameters

In this subchapter, several hyperparameters used during training will be explained. The hyperparameters contained in the model are as follows:

a. Word Embedding

The Word Embedding used is pretrained GloVe 300 dimensions which was trained with the 2014 Wikipedia website dataset and Gigaword fifth edition with a total of 6 billion tokens and pretrained Turian 50 dimensions which was trained from the RCV-1 Newswire dataset with a total of 40 million tokens.

b. Character Embedding

The Convolutional Network input vector for each character is 8 dimensions. The filters used are sizes 3, 4, and 5, and there are 50 of each.

c. Bidirectional LSTM

Each Fully Connected Layer in Bidirectional LSTM has 200 output nodes.

d. Learning Rate

The initial learning rate is 0.001. Decay is used to reduce the learning rate when there are more iterations. Decay Rate is 0.999 and Decay Frequency is 100, so decay is carried out every 100 iterations.

e. Dropout Rate

The large dropout value is divided into 2, namely Lexical Dropout and Dropout. Lexical Dropout is 0.5 and Dropout is 0.2. The larger the value, the greater the possibility of the node being discarded. Lexical Dropout is only used for word representation input, others use Dropout.

f. Hidden Layer Neural Network

End-to-End architecture uses 2 hidden layers with each layer having 150 hidden nodes.

g. Feature Size

All Feature Vectors used, from metadata, distance, and mention length all have an initial vector of 20 dimensions.

h. Constraints

The maximum length of sentences entered is 50. Each mention has a maximum of 250 antecedents. A mention ratio of 0.4 means that 40% of the total number of tokens will be mentioned. Mention width is 10, which means the longest span is formed from 10 tokens.

## 3.10 Optimizer

Optimizer in Neural Network is a method used to change Neural Network attributes such as weight in order to reduce loss values. The optimizer will run after getting the gradients of the learnable attributes that are searched through backpropagation.

All optimizations used in End-to-End Neural Coreference Resolution are based on Gradient Descent optimization. Gradient Descent is an optimization algorithm used to minimize the loss function by updating the weight repeatedly based on the gradient size.

**Algorithm 3 Algorithm Gradient Descent**

```
D[ITERATION]
FOR I=0 TO EPOCH-1
   G = ZEROS(W)
   FOR J=0 TO ITERATION-1
      G += CALC_GRAD(D[J])
   NEXT J
   UPDATE_WEIGHT(G)
```

Received: January 9, 2024. Revised: March 12, 2024.

```
NEXT I
```

Algorithm 3 is Algorithm Gradient Descent where ITERATION is the number of datasets, EPOCH is the number of times that will be repeated for the entire dataset, D is a list of datasets, G is a list of gradients of all weights, W is the number of weights, the ZEROS function is used to initialize a vector along parameters with all values being zero, the CALC_GRAD function function to calculate the gradient based on document-to-parameter input, and the UPDATE_WEIGHT function.

**Algorithm 4 Algorithm Stochastic Gradient Descent**

```
D[ITERATION]
FOR I=0 TO EPOCH-1
  FOR J=0 TO ITERATION-1
    G = CALC_GRAD(D[J])
    UPDATE_WEIGHT(G)
  NEXT J
NEXT I
```

Algorithm 4 is **Algorithm Stochastic Gradient** to update the weights immediately after the gradient is found from the input one dataset record. However, this algorithm can cause very long training if the dataset is very large, therefore a Mini-Batch Gradient Descent is formed to overcome this.

**Algorithm 5 Algorithm Mini-Batch Gradient Descent**

```
D[ITERATION]
FOR I=0 TO EPOCH-1
  BATCH[F][B] = CREATE_BATCH(D)
  FOR J=0 TO F-1
    FOR K=0 TO B-1
      G = CALC_GRAD(BATCH[J][K])
    NEXT K
    UPDATE_WEIGHT(G)
  NEXT J
NEXT I
```

Algorithm 5 is Mini-Batch Gradient Descent where BATCH is a list of batches that is formed containing as many batches as F and each batch has as many as B. CREATE_BATCH is a function used to separate a dataset into several batches.

### 3.11 Evaluation

This subchapter will explain how to evaluate Coreference Resolution. In the End-to-End Neural Coreference Resolution model, 3 types of metrics are used to determine the precision, recall, and F1-Score of the model, namely the standard metrics MUC, B3, and CEAFϕ4.

The MUC metric is a metric that measures the number of correct coreference links.

The Kuhn-Munkres algorithm is a technique used for solving large-scale assignment problems, such as matching workers to tasks or students to schools. It works by finding the optimal assignment that minimizes the total cost. However, this technique does have some drawbacks. For example, it can be slow or infeasible for very large or dynamic problems, which can require a lot of iterations and updates to find the optimal solution. It can also be sensitive to the choice of cost matrix, which can affect the quality and stability of the solution. Additionally, it can be limited by the assumption of one-to-one matching, which may not hold for some real-world problems that involve multiple or partial assignments[29].

**Algorithm 6 Algorithm Kuhn-Munkres**

```
M[COLUMN,ROW]
FOR I=0 TO ROW-1
  SUBTRACT_ROW(M,I)
NEXT I
FOR I=0 TO COLUMN-1
  SUBTRACT_COLUMN(M,I)
NEXT I
COVER_ZEROS(M)
WHILE ZERO_LINES(M)<ROW
  K=SUBTRACT_SMALLEST(M)
  ADD_COVER(K)
  COVER_ZEROS(M)
END WHILE
SELECT_MATCH(M)
```

Algorithm 6 is Algorithm Kuhn-Munkres where M is an RxK adjacency matrix whose value is negative, SUBTRACT_ROW is a function to reduce each row by the element with the smallest value in that row, while SUBTRACT_COLUMN does the same thing as SUBTRACT_ROW but in columns. COVER_ZEROS is a function used to cover matrix rows or columns that have a value of 0, and the number of closures is as minimal as possible.

**Algorithm 7 Algorithm Get Prediction and Loss**

```
FUNCTION GET_PREDICTION_AND_LOSS
  Initiate
  Input
  Bidirectional LSTM
  Span Representation
  Choose Mention
```

Received: January 9, 2024. Revised: March 12, 2024.

```
34:     Antecedent    Score    and
Coreference Score
35:     Loss
36:     Return
37:   END FUNCTION
```

Algorithm 7 is Algorithm to get Prediction and Loss.

## 4. Results and discussion

### 4.1 Dataset

The OntoNotes-5.0 dataset is the latest release of OntoNotes[30], an annotated corpus whose annotations are provided in separate text files for each annotation layer. This dataset has been widely used for Named Entity Recognition, Coreference Resolution, and Semantic Role Labeling tasks. This dataset is also the dataset most often used in Coreference Resolution research at this time and is also the dataset used in research on the main reference paper. Ontonotes-5.0 can be obtained from LDC free of charge for non-profit research and education.

The OntoNotes-5.0 dataset has 3 different languages, namely English, Chinese, and Arabic. Apart from that, the dataset is divided into 6 genres, namely Newswire, Broadcast News, Broadcast Conversation, Web Text, Telephone Conversation, and Pivot Corpus.
Table 8 is a statistic of the number of all tokens with columns as language and rows as genre. In the English dataset with the Newswire genre, 625 thousand tokens consist of 300 thousand Wall Street Journal tokens and 325 thousand tokens of the English part of the English-Chinese Parallel Treebank (ECTB).

The next dataset is a dataset created from the Book of Mark from the 1974 New Translation of the Bible. Each token of the Book of Mark will be tagged manually into CoNLL-2012 format. A document is a passage.

Table 8. Number of Ontonotes-5.0 Tokens by Language and Genre

| | **English** |
|---|---|
| Newswire | 625 K |
| Broadcast News | 200 K |
| Broadcast Conversation | 200 K |
| Web Text | 300 K |
| Telephone Conversation | 120 K |
| Pivot Corpus | 300 K |

Table 9. Datasets Statistic

| **Dataset** | **Token Train** | **Token Dev** |
|---|---|---|
| OntoNotes-5.0 English | 1.3 million (2802 document) | 160 thousand (343 document) |
| Book of Mark | 14.4 thousand (65 document) | 3.7 thousand (22 document) |

The Table 9 above is a table of many documents and their tokens in each set. There are 78 passages and a total of 18 thousand tokens in the Book of Mark. The 78 passages were divided into train and dev sets with a ratio of 75% for training and 25% for validation.

### 4.2 Data preprocessing

In machine learning, preprocessing is done to convert the dataset into a format suitable for the training model. This subchapter will explain 3 files that perform preprocessing of the dataset before running it on the main model. These files are minimize.py, get-char-vocab.py, and filter-embeddings.py.

#### 4.2.1 Minimize

The minimize.py file converts CoNLL format files to JSON. It extracts only the necessary information for the model. The Document State class is introduced to store the contents of a dataset, including document ID, sentences, and coreference clusters. The main functions involve checking and ensuring the correctness of the class, as well as transforming the final results into a JSON format that the model can use.

#### 4.2.2 Get char vocab

The get-char-vocab.py file is designed to create a file containing all unique characters across the entire dataset. It includes functions to generate a character vocabulary file based on the training, development, and test sets.

#### 4.2.3 Filter embeddings

To address the large size of the GloVe 300-dimensional file, filter-embeddings.py is employed to reduce the Word Embedding size. This file filters out words from the Word Embedding that do not appear in any of the datasets, resulting in a smaller Word Embedding file. The execution of this file involves specifying the Word Embedding file and the JSON dataset files for filtering.

#### 4.2.4 CoNLL-2012

Because the information from the OntoNotes dataset is split into several separate files, it will first be converted to CoNLL-2012 format. Instructions for conversion to CoNLL-2012 format can be accessed on the CoNLL-2012 website. The CoNLL-2012 format has 13 columns, namely as follows:

Document ID is the ID of the document, with the first 2 letters representing the genre of the document. The Broadcast News document genre is represented by the initials bn, Broadcast Conversation is represented by the initials bc, Web Text is represented by the initials wb, Telephone Conversation is represented by the initials tc, and Pivot Corpus is represented by the initials pt. Newswires in the Wall Street Journal and ECTB Xinhua section are represented by the initials nw, while Newswires in the ECTB magazine section are represented by the initials mz.

Part Number is an index that indicates the number of a document in a file.

Word Numbers is a sequence index of tokens starting from 0 in each sentence.

Word is a section that the actual token form in the document.

Part-of-Speech is a category of words based on their syntactic function such as nouns, verbs, pronouns, adjectives, and so on.

Bit Parse is the syntactic structure of the document, with the character * replaced by Part-of-Speech from column e which is a leaf of the Syntactic Tree.

Predicate Lemma contains the basic form of words for words that have Word Sense or Semantic Role. Additionally, it will contain "-".

Predicate Frameset ID contains the basic form of words for words that have Word Sense or Semantic Role. Additionally, it will contain "-".

Word Sense contains the Word Sense of the words in column d.

Speaker or Author contains the name of the person speaking or the person writing the token of the document.

Named Entities is a Named Entity of tokens or phrases, where a phrase is stated if it is contained in the same brackets, with * which can be replaced by the word in column d.

Predicate Arguments is a column for each argument predicate information structure for the predicates in column g.

Coreference is in the last column. This is information from a coreference that has a phrase system with brackets, where the phrase from a token that has a coreference ID tag with an open bracket goes to a token that has a coreference ID tag with a closed bracket. Phrases that have a coreference relationship are phrases that have the same number in a document.

### 4.3 Experiments

The model was evaluated using the MUC, $B^3$, and CEAF $_{\phi 4}$ metrics . The MUC metric focuses on coreference links, $B^3$ focuses on mentions in the coreference cluster, while CEAF $_{\phi 4}$ focuses on a cluster of pairs that are most similar. The final F1 is the average of the three metrics.

Training is conducted on the English and Indonesian models for 150 epochs and the model will be evaluated every 5 epochs. ADAM is used as optimizer. Given a dropout of 0.5 on the Word representation and Character Embedding and 0.2 on the hidden layer and on the Feature Vector.

Constraint sentence in a document is 50, the maximum number of antecedents is 250, and the maximum span length is 10.

The ANN used in the Mention Score and Antecedent Score has a hidden layer 2, with 150 nodes for the English model and 100 nodes for the Indonesian model.

Word Representation uses Word Embedding which is described in III while the Indonesian Word Embedding is 100 vector sizes made with GloVe with the entire book of Mark dataset. Character Embedding[31] uses 3 types of filters with sizes [3, 4, 5] and each filter has 50, while the initial representation of the characters themselves has dimensions of 8.

pruning is carried out until there is a span of 40% of the number of tokens remaining. In the UK model, this means that the model only uses an average of 4% of the formed span. However, the model still manages to get 92% correct mentions.

#### 4.3.1 English model

Fig. 12 shows that based on the recall and precision obtained, the model becomes bolder in

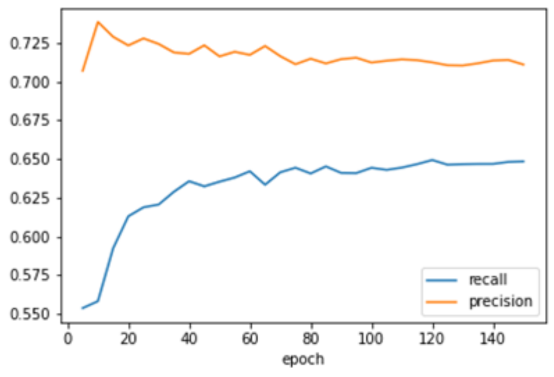


Figure. 12 Recall dan Precision from English Model

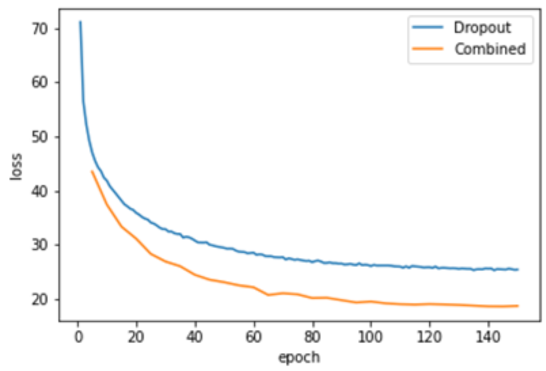


Figure. 13 Loss Graph with Dropout and without Dropout on English Model

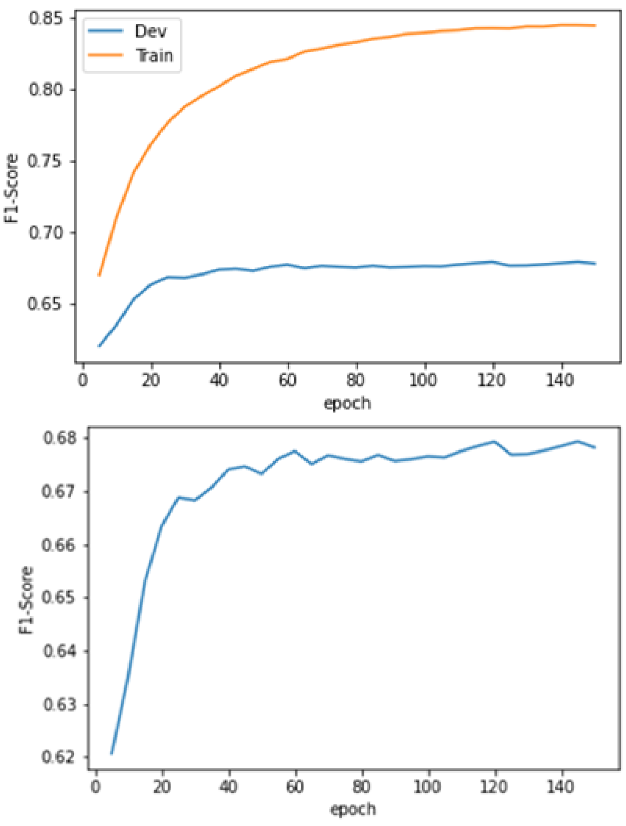


Figure. 14 Comparison of F1-Score Set Train and Dev on English Model

guessing the coreference the more epochs are trained.

Fig. 13 shows that the resulting loss calculated with some nodes dropped out is much greater than when all nodes are used. This proves that dropout functions well because the resulting loss when combined is actually very small even though the node selection is different in each iteration.

Fig. 14 shows that the image on the bottom is an enlargement of the "Dev" graph on the left. The results of the training set's F1-Score continued to increase well every epoch up to 85%, while the evaluation results had a much smaller F1-Score. However, the evaluation results still tend to increase the F1-Score for each epoch.

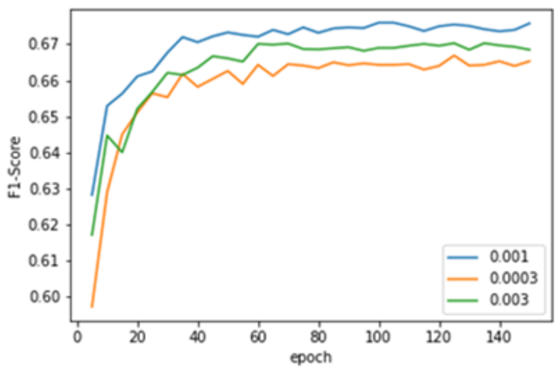


Figure. 15 Tuning F1-Score Learning Rate on English Model

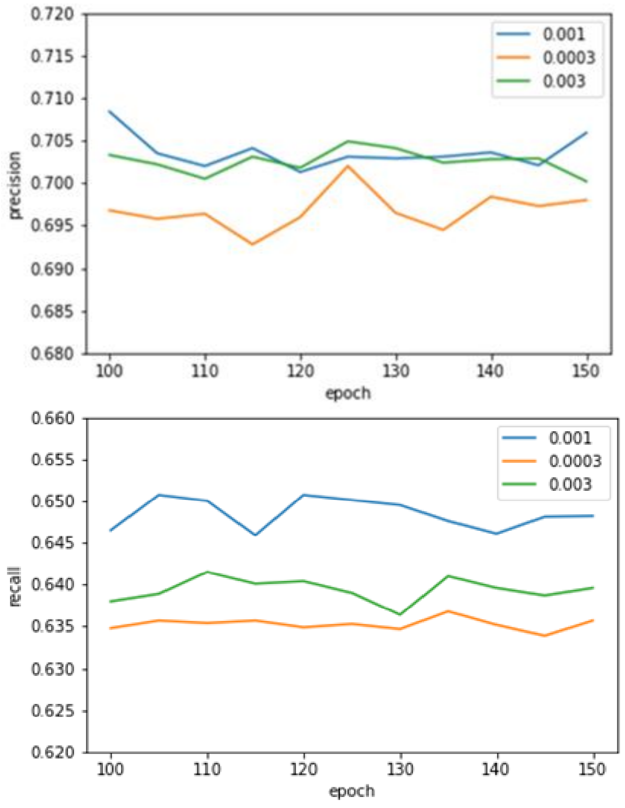


Figure. 16 Precision dan Recall Tuning Learning Rate on English Model

Fig. 15 shows that the first tuning carried out is the learning rate. The best results are still at the default learning rate of 0.001.

Fig. 16 shows that there is no significant difference in both recall and precision. However, a learning rate of 0.001 still has the best precision and recall compared to other learning rates.

Fig. 17 shows that the best number of LSTM nodes is 150. The default hyperparameter of 200 has smaller results with an F1-Score of 67.59%. The F1-Score of 250 nodes has stopped growing after 85 epochs while underfitting occurs on 100 nodes.

Fig. 18 shows that there is not much difference in the recall values while the precision values for 100

Received: January 9, 2024. Revised: March 12, 2024.

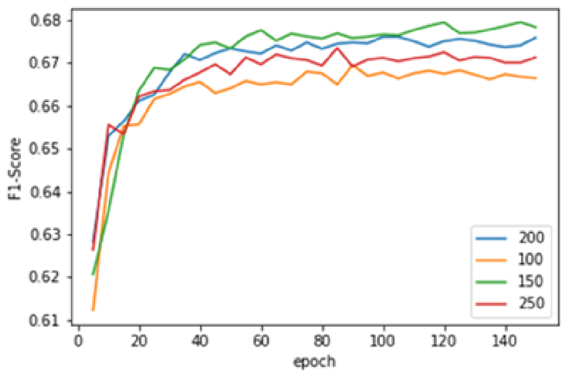


Figure. 17 Tuning F1-Score LSTM Node on English Model

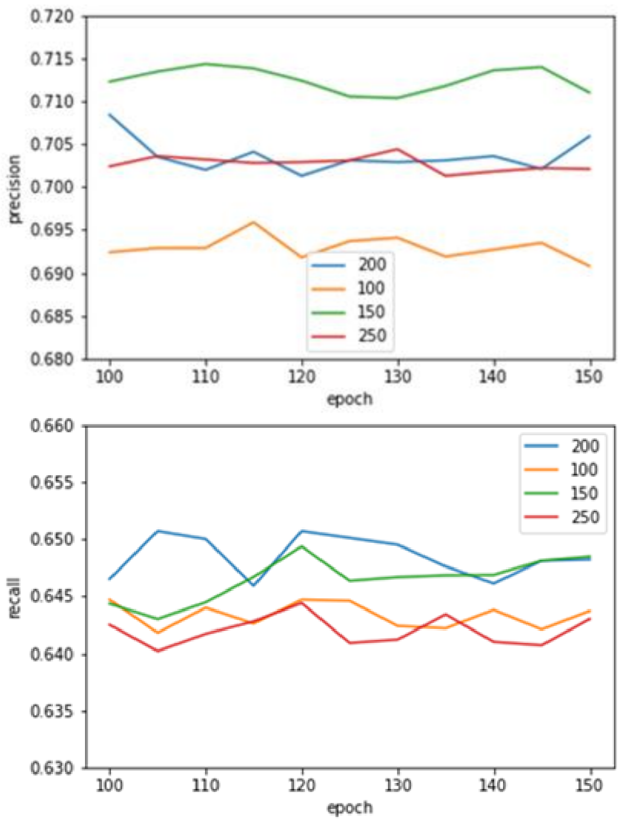


Figure. 18 Precision dan Recall Tuning LSTM Node on English Model

nodes are much lower while the precision values for 200 nodes and 250 nodes are not much different from the best. The results of 150 nodes managed to beat other models, especially in the value of precision.

### 4.3.2 Mention score and antecedent score on English model

This subchapter explains the results of the Mention Score and Antecedent Score. Before getting a mention, spans are selected with 40% of the document tokens from the spans that have the highest Mention Score. So, because the longest mention limit is 10 and the OntoNotes-5.0 dataset has an average number of tokens of 467, a maximum of 4625 spans will be formed if there are 467 tokens in the document, even though only 40% of the total number of tokens, namely 187 spans, are selected. This means that in documents the average span size selected is only 4% of the total number of spans. Remaining spans that are not used will not receive updates in the backward pass. Even though it only used 4%, the models that were tried succeeded in getting 92% of mentions since the first 5 epochs which were not prune except for models that did not use LSTM and Word Embedding. Even though it didn't use LSTM and Word Embedding, the model still managed to reach 82% mentions in the first 5 epochs, and rose to 86%

Table 10. Mention Score and Antecedent Score Table

Doc = Donald is the president of USA which is a country in North America.
That country has a lot of states.
The president is older than me.

| Index | Mention | Mention Score |
|---|---|---|
| 0 | Donald | 0.44 |
| 1 | USA which is a country | -3.7 |
| 2 | USA which is a country in North America | 0.66 |
| 3 | North America | 0.34 |
| 4 | That country | 0.76 |
| 5 | have | -1.46 |
| 6 | a lot of states | 0.36 |
| 7 | The president | 0.78 |
| 8 | older | -1.42 |
| 9 | older than me | -3.76 |
| 10 | me | 0.3 |

**Antecedent "That Country"**

| Index Mention | Antecedent Score | Coreference Score |
|---|---|---|
| 0 | -8.57 | -7.37 |
| 1 | 2.04 | -0.91 |
| 2 | 3.71 | 5.13 |
| 3 | 3.21 | 4.31 |
| ϵ | - | 0 |

Table 11. Mention Score for All Subjects and Some Subjects

Doc = Lisa and John were friends.
They met in a small town.
She was a few years older than him.

| Mention | Mention Score |
|---|---|
| Lisa and John | 0.52 |
| John | -1.17 |
| were | -1.67 |
| They | 0.37 |
| met | -0.71 |
| a small town | 0.72 |
| She | 0.52 |
| him | 0.19 |

Table 12. The Complicated Problem of Coreference

Doc 1 = James was robbed by George and now he is in prison.

Doc 2 = Felicia was robbed by George and now he is in prison.

**Antecedent "he" Doc 1**

| Antecedent | Antecedent Score |
|---|---|
| James | 7.66 |
| robbed | -3.54 |
| George | 7.24 |

**Antecedent "he" Doc 2**

| Antecedent | Antecedent Score |
|---|---|
| Felicia | 2.88 |
| robbed | -3.96 |
| George | 6.68 |

when it reached 30 epochs. However, the model without these two features gets an F1-Score of 43.76% in 30 epochs, while with these two features the model gets an F1-Score of 66.87% in the same number of epochs.

The Mention Score and Antecedent Score values have quite a wide range when compared, table 10 is an example of model results without metadata features. In Table 10, the top table is the spans selected as mentions while the right table is the antecedents of the mention "That country". It can be seen that the Mention Score tends to be much smaller in value when compared to the Antecedent Score. This means that the role of the Antecedent Score is much more important than the Mention Score when determining antecedents. It can also be seen that the Mention Score is greater for words that are full subjects or objects, which means that the LSTM used in the model allows the model to detect patterns between nouns and conjunctions or verbs. Based on Table 11, phrases that are a whole subject such as "Lisa and John" have a good Mention Score while words that are a small part of a subject such as "John" actually have a bad Mention Score, even the word "Lisa" is not included in the mentions. The model tends to detect full Noun Phrases. This can be good for detecting full names, but bad in instances of more than 1 subject like the example above. However, the model can still detect phrases that have pronouns such as "his" in "his children" well. This is also one of the disadvantages of combining learning to search for mentions or entities which is usually done with Named Entity Recognition directly with antecedent searches, because even though a span is a Named Entity, it will still be considered to have negative feedback on the model if it happens to have no antecedents. Furthermore, in Table 12 there are 2 examples of documents with Doc 1 being a fairly complicated coreference problem:

In Table 12, the tricky problem in Doc 1 is determining whether "he" is "George" or "James", where the model is wrong because it prefers "James". Models tend to choose the earliest nouns in problems like this. In Doc 2, because the detected genders are different, the model can easily determine the correct coreference relationship.

### 4.3.3 Indonesian model

Fig. 19 shows the "normal" line is the F1-Score obtained every 2 epochs. It can be seen that each epoch has quite a large difference in F1-Score and that is why evaluations are carried out every 2 epochs so that we don't easily lose the best results.

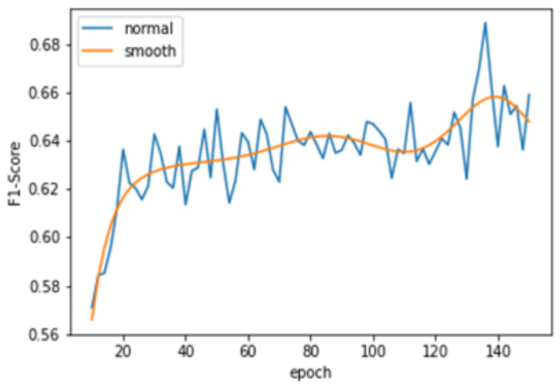


Figure. 19 F1-Score Evaluation on Indonesian Model

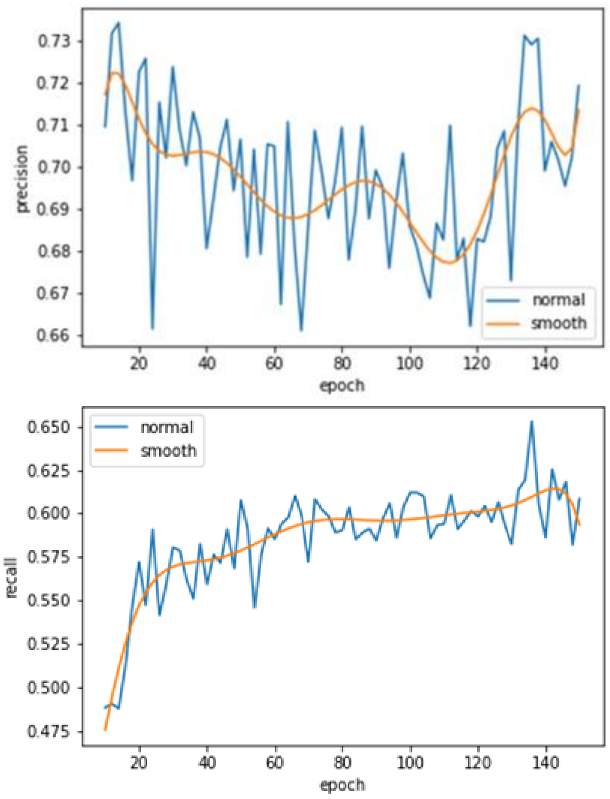


Figure. 20 Precision and Recall on Indonesian Model

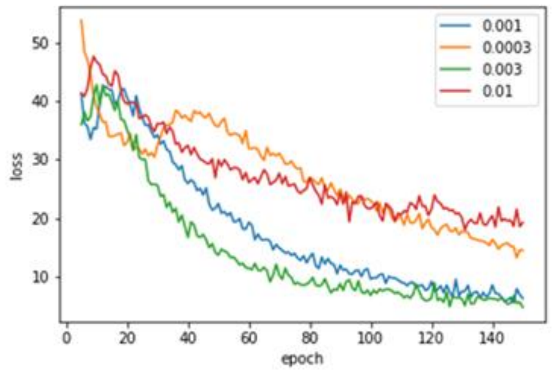


Figure. 21 Tuning Loss Learning Rate on Indonesian Model

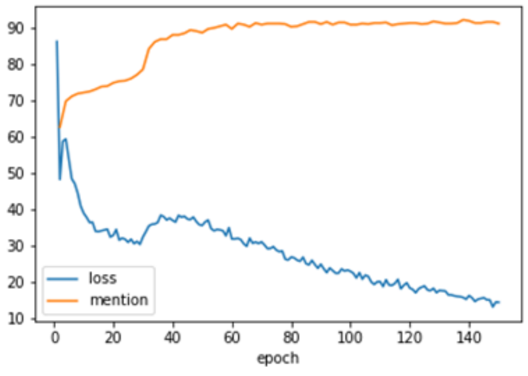


Figure. 22 Detected Mentions and Loss

The F1-Score value still tends to increase every epoch. The "smooth" line is a "normal" line that is smoothed

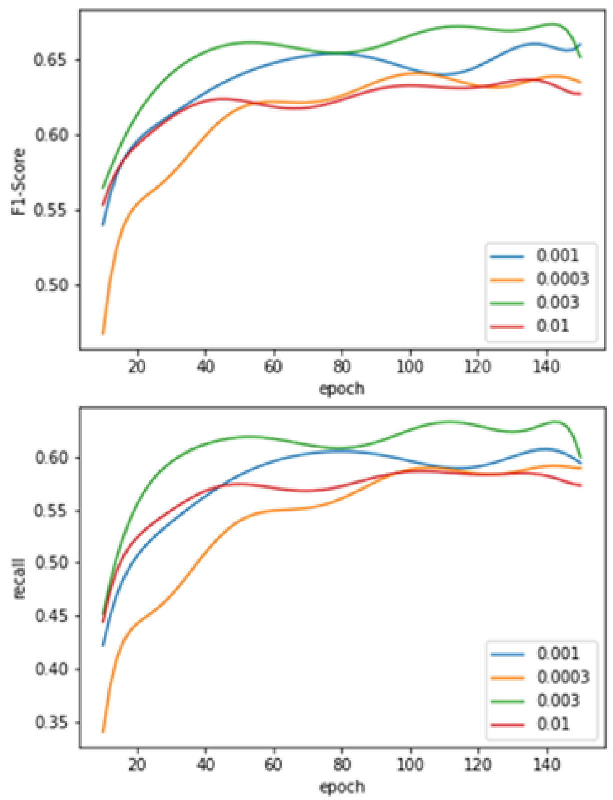


Figure. 23 F1-Score dan Recall Learning Rate Indonesia

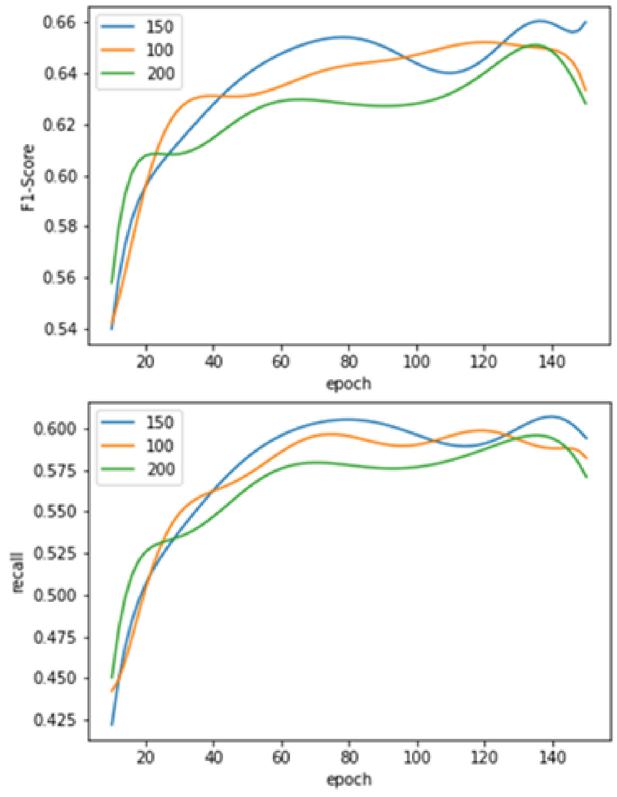


Figure. 24 F1-Score and Recall LSTM Node Indonesia

so that the difference with other lines can be seen easily which will be applied when tuning.

Fig. 20 shows the precision obtained is not stable every epoch. In the tuning carried out, the precision

Received: January 9, 2024. Revised: March 12, 2024.

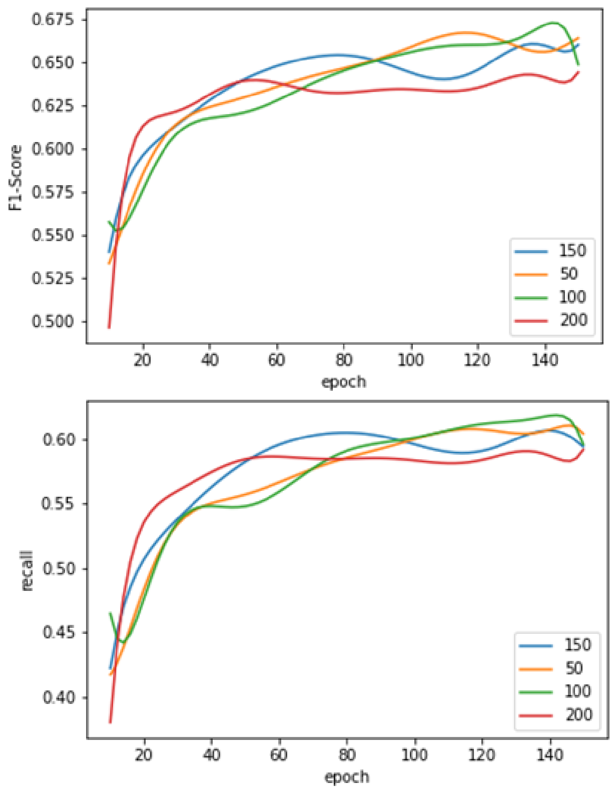


Figure. 25 F1-Score and Recall Hidden Node Indonesia

values for each different set of hyperparameters get different patterns as well. However, recall still tends to increase every epoch, just like in the UK case. These unstable results are due to the small number of datasets in Indonesian with a comparison of the training dataset of 1:43 and the evaluation dataset of 1:16 for the number of Indonesian and English documents.

Fig. 21 shows that the loss from learning rates of 0.003 and 0.001 both work well, while the loss of learning rate 0.0003 takes too long to converge, while the learning rate of 0.01 is difficult to reduce loss because the learning rate is too large. Then, it can be seen that the loss in the early epochs was unstable, and only stabilized down since several epochs were running.

Fig. 22 shows the new loss graph will stabilize downwards when the model detects mentions that are approximately 90% or more unprune. This is because the prune spans are not used in calculating loss.

Fig. 23 shows the F1-Score and recall values are quite identical in development with the 0.003 model having the best results.

Fig. 24 shows that F1-Score and recall metrics from 150 nodes get the best results while 100 nodes are still underfit while 200 nodes are overfitting.

Fig. 25 shows that it doesn’t take a lot of Hidden Nodes to get good results on the Indonesian set. The best performing results are those with 50 and 100 nodes. However, the development of 50 nodes stopped at 110 epochs while 100 nodes continued to grow.

## 4.4 Comparative analysis

### 4.4.1 English

The English Case Study uses the OntoNotes-5.0 dataset, train set for training and dev set for evaluation. The model that has the best evaluation results will be tested on the test dataset to become the final F1-Score. Training is carried out for up to 150 epochs. The default training hyperparameters are the parameters used in the reference paper. Because Google Colab sessions last a maximum of 12 hours, training is carried out every 30 epochs. Training of 30 epochs runs for approximately 3.5 hours on a Tesla P100-PCIE-16GB GPU and approximately 7 hours on a Tesla K80, which means it takes 17.5 hours to 35 hours to run 150 epoch training.

Table 13. English Results Without Certain Features

| | **F1** | **ΔF1** |
|---|---|---|
| Best | 67.94 | |
| - Word Embedding | 62.47 | -5.47 |
| - Char Embedding | 66.20 | -1.74 |
| - Span Head | 65.87 | -2.07 |
| - Feature Distance | 63.75 | -4.19 |
| - Feature Width | 67.35 | -0.59 |
| - Metadata Genre | 67.04 | -0.90 |
| - Metadata Speaker | 66.53 | -1.41 |
| | **P** | **ΔP** |
| Best | 71.24 | |
| - Word Embedding | 68.05 | -3.19 |
| - Char Embedding | 68.24 | -3.00 |
| - Span Head | 69.07 | -2.17 |
| - Feature Distance | 67.01 | -4.23 |
| - Feature Width | 70.30 | -0.94 |
| - Metadata Genre | 70.01 | -1.23 |
| - Metadata Speaker | 68.79 | -2.45 |
| | **R** | **ΔR** |
| Best | 64.93 | |
| - Word Embedding | 57.79 | -7.14 |
| - Char Embedding | 64.27 | -0.66 |
| - Span Head | 62.96 | -1.97 |
| - Feature Distance | 60.81 | -4.12 |
| - Feature Width | 64.64 | -0.29 |
| - Metadata Genre | 64.31 | -0.62 |
| - Metadata Speaker | 64.42 | -0.51 |

Table 14. Evaluation on English dataset with baseline model [15]

| Metric | Value |
|---|---|
| MUC | 0.51 |
| CEAF | 2.07 |
| BCubed | 0.83 |

All models are evaluated every 5 epochs. First, we will explain the best results obtained, then after getting the best results we will analyze the usefulness of the features used, then we will explain the hyperparameter tuning that was carried out to get the best results. After that, several other analyzes of the model will be explained.

The best results from the English case study were using the default hyperparameters with a combination of 150 LSTM layer nodes with an F1-Score of 67.94% on the dev set and a final successful F1-Score of 67.14% on the test set.

Table 13 shows the English Results Without Certain Features

Word Embedding causes the model to be able to handle more types of coreferring noun phrases that exist in the Word Embedding vocabulary. In addition, this feature also helps determine the antecedent of a pronoun based on gender.

Character Embedding has the advantage of providing word representations for words that are not contained in the vocabulary. Words that are outside the vocabulary tend to be guessed as coreferences to one another if the input representation is only Word Embedding because the representations are the same.

the Span Head is not used, the Bidirectional LSTM representation used is only the first and last words of the span. This causes information from the words in the middle of the span to be lost.

Distance is not used, pronouns that fail to find antecedents can actually look for antecedents to mentions that are very far apart because they do not know how big the distance between the two mentions is.

The speaker helps group the pronouns "I" and "you" where if the speakers are the same, then what can be grouped is "I" with "I" and "you" with "you". In addition, if the speakers are different, it raises the possibility that "you" and "I" can be grouped together.

It was also found that the model is more concerned with the Antecedent Score than the Mention Score. In addition, the model tends to be concerned with detecting full Noun Phrases in documents.

The coreference resolution system on Ontonotes dataset with the baseline model [15] achieved moderate performance with an MUC score of 0.51. However, there is room for improvement, as indicated by the CEAF score of 2.07. On the positive side, the system demonstrated relatively good precision and recall with a BCubed score of 0.83. Further analysis and optimization are recommended to enhance overall effectiveness.

### 4.4.2 Indonesian

Indonesian Case Study using the Book of Mark dataset. Training is carried out for up to 150 epochs. The default training hyperparameters are the same as the best English hyperparameters, only with num_filters Character Embedding of 30 and decay_frequency of 65 because the model has a much smaller number of datasets. Training 150 epochs runs for approximately 15 minutes on the Tesla P100 GPU and 25 minutes on the Tesla K80 GPU. All models are evaluated every 2 epochs.

The best results for the Indonesian language case study were getting an F1-Score of 68.88% with default hyperparameters with a combination of 100 Hidden Nodes and a learning rate of 0.003. The dropout effect and F1-Score results from the training set have the same pattern as the English case. Table 15 shows the Indonesia Results Without Certain Features.

Table 15. Indonesia Results Without Certain Features

| | **F1** | **ΔF1** |
|---|---|---|
| Best | 68.88 | |
| - Word Embedding | 63.36 | -5.52 |
| - Char Embedding | 58.89 | -9.99 |
| - Span Head | 65.44 | -3.44 |
| - Feature Distance | 66.60 | -2.28 |
| - Feature Width | 67.72 | -1.16 |
| | **P** | **ΔP** |
| Best | 72.91 | |
| - Word Embedding | 72.39 | -0.52 |
| - Char Embedding | 75.13 | +2.22 |
| - Span Head | 72.77 | -0.14 |
| - Feature Distance | 73.57 | +0.66 |
| - Feature Width | 72.79 | -0.12 |
| | **R** | **ΔR** |
| Best | 65.31 | |
| - Word Embedding | 56.52 | -8.79 |
| - Char Embedding | 48.70 | -16.61 |
| - Span Head | 59.70 | -5.61 |
| - Feature Distance | 60.93 | -4.38 |
| - Feature Width | 63.45 | -1.86 |

Table 16. Evaluation on Indonesian dataset with baseline model

| Dataset | Metric | Recall (R) | Precision (P) | F₁ Score |
|---|---|---|---|---|
| Original | MUC | 0.758 | 0.761 | 0.756 |
| | B³ | 0.627 | 0.538 | 0.543 |
| | CEAF_m | 0.456 | 0.495 | 0.475 |
| | CEAF_e | 0.379 | 0.439 | 0.389 |
| | BLANC | 0.607 | 0.598 | 0.551 |
| Our Dataset | MUC | 0.477 | 0.404 | 0.430 |
| | B³ | 0.885 | 0.619 | 0.697 |
| | CEAF_m | 0.621 | 0.621 | 0.621 |
| | CEAF_e | 0.557 | 0.775 | 0.624 |
| | BLANC | 0.773 | 0.795 | 0.760 |

Word Embedding allows important words that rarely appear in documents to have a greater chance of being mentioned.

Character Embedding is not used, the model loses recall of 16.61%, which means the model is underfit. In contrast to Word Embedding, which has a fixed value, Character Embedding is more flexible because Character Embedding is trained directly with a dataset task, while Word Embedding is trained regardless of what task will be performed.

Distance in Indonesian does not get a much higher performance because on average each Bible passage does not have many kinds of mentions in it so that the possibility of saying a coreference is smaller.

Despite getting pretty good results at the time of evaluation, the model still has difficulties in solving problems other than anaphora. The average document dataset created has a few clusters but many coreference relationships.

Even though there are many shortcomings in the model as described above, the Indonesian language model can work well in the domain of the book of Mark and does not need to require additional dataset processing or further architecture other than separating the affixes "*nya*", "*ku*", and "*mu*" on dataset creation.

The original dataset achieved respectable results across various evaluation metrics. For the MUC metric, the recall was 0.758, precision stood at 0.761, and the F₁ score reached 0.756. However, for B³, CEAF_m, CEAF_e, and BLANC, the performance was comparatively lower. These metrics ranged from 0.379 to 0.627 for recall, 0.439 to 0.538 for precision, and 0.389 to 0.543 for the F₁ score.

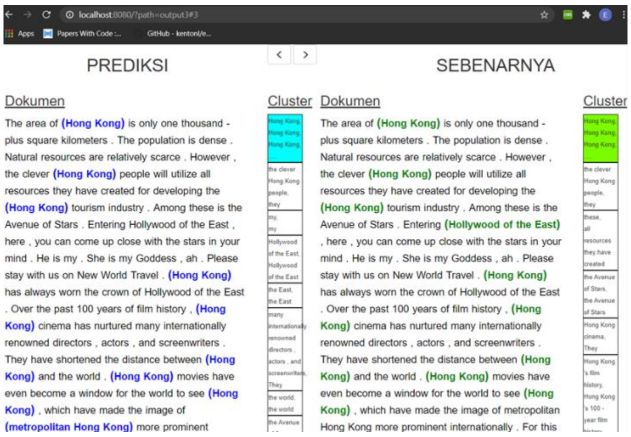
In contrast, our dataset exhibited different strengths and weaknesses. The MUC metric achieved a recall of 0.477 and a precision of 0.404, resulting in an F₁ score of 0.430. Notably, the B³ metric performed exceptionally well, with a recall of 0.885, precision of 0.619, and an impressive F₁ score of 0.697. Additionally, CEAF_m maintained consistent recall and precision at 0.621, while CEAF_e achieved a high precision of 0.775 and an F₁ score of 0.624. Lastly, BLANC demonstrated strong overall performance, with a recall of 0.773, precision of 0.795, and an F₁ score of 0.760[16].

## 4.5 Coreference visualization

To display the coreference results obtained from the model, visualization is carried out. There are 2 visualization media created, one for comparing predicted and actual coreference, and the other for carrying out direct testing from the input provided.

The following coreference visualization was created using HTML and Javascript. All that is needed to perform the following visualization is the JSON output from decoder.py. The following visualization is a modification of the main.js and index.html files in the original End-to-End Coreference Resolution source.

Fig. 26 is an example of a comparison of predicted and actual coreference clusters. In the cluster column, each box indicates a cluster and the contents of the box are a maximum of 3 spans of the clusters in the coreference. The "<" and ">" buttons at the top center are used to move to another document. In the URL, there is a parameter whose contents are the name of the JSON file that will be visualized and next to it is "#x" where x is the index of the document.



PREDIKSI — Dokumen

The area of (Hong Kong) is only one thousand - plus square kilometers . The population is dense . Natural resources are relatively scarce . However , the clever (Hong Kong) people will utilize all resources they have created for developing the (Hong Kong) tourism industry . Among these is the Avenue of Stars . Entering Hollywood of the East , here , you can come up close with the stars in your mind . He is my . She is my Goddess , ah . Please stay with us on New World Travel . (Hong Kong) has always worn the crown of Hollywood of the East . Over the past 100 years of film history , (Hong Kong) cinema has nurtured many internationally renowned directors , actors , and screenwriters . They have shortened the distance between (Hong Kong) and the world . (Hong Kong) movies have even become a window for the world to see (Hong Kong) , which have made the image of (metropolitan Hong Kong) more prominent

PREDIKSI — Cluster

- Hong Kong, Hong Kong, Hong Kong, ...
- the clever Hong Kong people, they
- my, my
- Hollywood of the East, Hollywood of the East
- the East, the East
- many internationally renowned directors , actors , and screenwriters, They
- the world, the world
- the Avenue

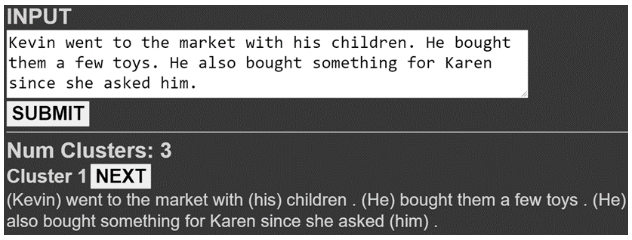
SEBENARNYA — Dokumen

The area of (Hong Kong) is only one thousand - plus square kilometers . The population is dense . Natural resources are relatively scarce . However , the clever (Hong Kong) people will utilize all resources they have created for developing the (Hong Kong) tourism industry . Among these is the Avenue of Stars . Entering (Hollywood of the East) , here , you can come up close with the stars in your mind . He is my . She is my Goddess , ah . Please stay with us on New World Travel . (Hong Kong) has always worn the crown of Hollywood of the East . Over the past 100 years of film history , (Hong Kong) cinema has nurtured many internationally renowned directors , actors , and screenwriters . They have shortened the distance between (Hong Kong) and the world . (Hong Kong) movies have even become a window for the world to see (Hong Kong) , which have made the image of metropolitan Hong Kong more prominent internationally . For this

SEBENARNYA — Cluster

- Hong Kong, Hong Kong, Hong Kong, ...
- the clever Hong Kong people, they
- these, all resources they have created
- the Avenue of Stars, the Avenue of Stars
- Hong Kong cinema, They
- Hong Kong 's film history, Hong Kong 's 100 - year film

Figure. 26 Comparative Coreference Visualization


INPUT

Kevin went to the market with his children. He bought them a few toys. He also bought something for Karen since she asked him.

SUBMIT

Num Clusters: 3

Cluster 1 NEXT

(Kevin) went to the market with (his) children . (He) bought them a few toys . (He) also bought something for Karen since she asked (him) .

Figure. 27 Interface Testing Coreference

## 4.6 Web application

The IPython library is used to create the interface for the web application. In Fig. 27, the required input is a document written in the textarea. After you have finished writing the document, click the "SUBMIT" button to get the cluster prediction results.

The output result is "Num Clusters" as how many clusters are formed and visualization of a document according to a cluster. The "NEXT" button functions to move to another cluster visualization.

## 4.7 Discussion

The End-to-End Neural Coreference Resolution model comes with a host of advantages. It eliminates the need for complicated preprocessing during both training and testing phases. The model is versatile and can be applied to other languages with minimal additional preprocessing. The process of applying coreference resolution is streamlined into a single stage of model creation. It has the potential to outperform previous studies by utilizing only the features of the actual text and its metadata. Furthermore, it eliminates the need for calculations across the entire span that is formed.

However, the model also has its share of disadvantages. It demands large datasets from various domains. As it is integrated into a Neural Network architecture, the mentions learned are not name entities, but rather entities that have more frequent coreferences in the dataset. The model still struggles with complex coreference problems. Tracing to identify issues related to coreference resolution in the model is challenging due to its Neural Network-based nature.

## 5. Conclusion

The experiments conducted led to several conclusions. The English model was found to detect coreference with an F1-Score of 67.14% in the OntoNotes-5.0 dataset, while the Indonesian model achieved an F1-Score of 68.88%. Despite the higher F1 score of the Indonesian model, the English model performed better on independent datasets due to its larger dataset and different domain.

LSTM played a crucial role in determining the span which is a mention, as it could detect patterns of sentence parts. The model was able to determine 92% of the correct mentions, even though it selected only approximately 4% of all spans, pruning the remaining 96% of the span.

All additional features, except for the long mention feature, provided a significant performance boost to the model. The long mention feature only increased the performance of the British model by 0.59%, while other features added at least 0.9%. Mention length contributed to the performance of the Indonesian model by 1.16%, while other features added at least 2.28%.

Character Embedding emerged as a good text alternative feature due to its flexibility and adaptability to the task at hand. Without using Word Embedding and only Character Embedding in the input, the British model achieved an F1 of 62.47%, while the Indonesian model achieved an F1 of 63.36%. Word Embedding remained a superior feature on large datasets, but Character Embedding alone managed to yield satisfactory results.

It was found unnecessary to initialize the LSTM weights with an orthogonal matrix, as LSTM itself could handle the problem of vanishing gradients and exploding gradients well. The initial processing to form the Indonesian language model only required the separation of the affixes "*nya*", "*ku*", and "*mu*" when forming the dataset.

A dataset from just one domain performed quite well in that domain but faltered when applied to other domains due to a lack of vocabulary and bias towards the text pattern of that domain only. End-to-End Neural Coreference Resolution models were found to require large datasets on Word Embedding training and the model itself, and from several different domains, to achieve better results.

## Conflicts of Interest

The authors declare no conflict of interest.

## Author Contributions

Conceptualization, Gunawan, Esther I. Setiawan, Ivan Hosea; methodology, Gunawan, Esther I. Setiawan, Ivan Hosea, Kimiya Fujisawa; software, Ivan Hosea; validation, Gunawan, Ivan Hosea; formal analysis, Esther I. Setiawan and Kimiya Fujisawa; writing—original draft preparation, Gunawan, Ivan Hosea, Esther I. Setiawan; writing—review and editing, Gunawan, Ivan Hosea, Esther I. Setiawan and Kimiya Fujisawa ; visualization, Ivan Hosea; supervision, Gunawan, Esther I. Setiawan and Kimiya Fujisawa; project administration, Gunawan,

Esther I. Setiawan and Kimiya Fujisawa; funding acquisition, Gunawan.

## References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning", In: *Proc. of the 26th Annual International Conference on Machine Learning*, pp. 41-48, 2009.

[2] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges", *Multimed Tools Appl*, Vol. 82, No. 3, pp. 3713-3744, 2023.

[3] W. M. Soon, H. T. Ng, and D. C. Y. Lim, "A machine learning approach to coreference resolution of noun phrases", *Computational linguistics*, Vol. 27, No. 4, pp. 521-544, 2001.

[4] E. I. Setiawan, J. Santoso, and others, "Answer Ranking with Weighted Scores in Indonesian Hybrid Restricted Domain Question Answering System", In: *Proc. of 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, pp. 456-461, 2021.

[5] E. I. Setiawan, H. Juwiantho, J. Santoso, S. Sumpeno, K. Fujisawa, and M. H. Purnomo, "Multiview Sentiment Analysis with Image-Text-Concept Features of Indonesian Social Media Posts", *International Journal of Intelligent Engineering & Systems*, Vol. 14, No. 2, 2021, doi: 10.22266/ijies2021.0430.47.

[6] K. Clark and C. D. Manning, "Improving coreference resolution by learning entity-level distributed representations", *arXiv preprint arXiv:1606.01323*, 2016.

[7] S. Wiseman, A. M. Rush, and S. M. Shieber, "Learning global features for coreference resolution", *arXiv preprint arXiv:1604.03035*, 2016.

[8] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes", In: *Proc. of Joint conference on EMNLP and CoNLL-shared task*, pp. 1-40, 2012.

[9] J. Santoso, E. I. Setiawan, C. N. Purwanto, E. M. Yuniarno, M. Hariadi, and M. H. Purnomo, "Named entity recognition for extracting concept in ontology building on Indonesian language using end-to-end bidirectional long short term memory", *Expert Syst Appl*, Vol. 176, p. 114856, 2021.

[10] K. Chowdhary and K. R. Chowdhary, "Natural language processing", *Fundamentals of artificial intelligence*, pp. 603-649, 2020.

[11] T. M. Lai, T. Bui, and D. S. Kim, "End-to-end neural coreference resolution revisited: A simple yet effective baseline", In: *Proc. of ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8147-8151, 2022.

[12] J.-C. Gu, Z.-H. Ling, and N. Indurkhya, "A study on improving end-to-end neural coreference resolution", In: *Proc. of Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 17th China National Conference, CCL 2018, and 6th International Symposium, NLP-NABD 2018, Changsha, China, October 19-21, 2018, Proceedings 17*, pp. 159-169, 2018.

[13] C. Park, J. Lim, J. Ryu, H. Kim, and C. Lee, "Simple and effective neural coreference resolution for Korean language", *ETRI Journal*, Vol. 43, No. 6, pp. 1038-1048, 2021.

[14] T. Auliarachman and A. Purwarianti, "Coreference Resolution System for Indonesian Text with Mention Pair Method and Singleton Exclusion using Convolutional Neural Network", In: *Proc. of 2019 International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pp. 1-5, 2019.

[15] V. Dobrovolskii, "Word-level coreference resolution", *arXiv preprint arXiv:2109.04127*, 2021.

[16] V. K. P. Artari, R. Mahendra, M. A. Jiwanggi, A. Anggraito, and I. Budi, "A multi-pass sieve coreference resolution for Indonesian", In: *Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pp. 79-85, 2021.

[17] V. K. P. Artari, R. Mahendra, M. A. Jiwanggi, A. Anggraito, and I. Budi, "A multi-pass sieve coreference resolution for Indonesian", In: *Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pp. 79-85, 2021.

[18] G. J. Suherik and A. Purwarianti, "Experiments on coreference resolution for Indonesian language with lexical and shallow syntactic features", In: *Proc. of 2017 5th International Conference on Information and Communication Technology (ICoIC7)*, pp. 1-5, 2017.

[19] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution", *arXiv preprint arXiv:1707.07045*, 2017.

[20] K. Clark and C. D. Manning, "Deep reinforcement learning for mention-ranking coreference models", *arXiv preprint arXiv:1609.08667*, 2016.

[21] H. Cui, J. Zhang, C. Cui, and Q. Chen, "Solving large-scale assignment problems by Kuhn-Munkres algorithm", In: *Proc. of 2nd Int. Conf. Advances Mech. Eng. Ind. Inform. (AMEII 2016)*, 2016.

[22] P. Elango, "Coreference resolution: A survey", *University of Wisconsin, Madison, WI*, p. 12, 2005.

[23] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation", In: *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543, 2014.

[24] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning", In: *Proc. of the 48th annual meeting of the association for computational linguistics*, pp. 384-394, 2010.

[25] M. Rivai, F. Kurniawan, and others, "Diabetes Detection Using Carbon Nanomaterial Coated QCM Gas Sensors and a Convolutional Neural Network through Urine Sample", *International Journal of Intelligent Engineering & Systems*, Vol. 16, No. 5, 2023, doi: 10.22266/ijies2023.1031.36.

[26] S. Patro, J. Mishra, and B. S. Panda, "Hybrid Convolutional Neural Network with Residual Neural Network for Breast Cancer Prediction Using Mammography Images", *International Journal of Intelligent Engineering & Systems*, Vol. 16, No. 1, 2023, doi: 10.22266/ijies2023.0228.33.

[27] S. Vikkurty and P. Setty, "Artificial Neural Network based Optimized Link State Routing Protocol in MANET", *International Journal of Intelligent Engineering & Systems*, Vol. 15, No. 6, 2022, doi: 10.22266/ijies2022.1231.07.

[28] M. S. Sawah, S. A. Taie, M. H. Ibrahim, and S. A. Hussein, "Deep Neural Network-based Approach for Accurate Vehicle Counting", *International Journal of Intelligent Engineering & Systems*, Vol. 16, No. 2, 2023, doi: 10.22266/ijies2023.0430.21.

[29] M. Mayez, K. Nagaty, and A. Hamdy, "Developer Load Normalization Using Iterative Kuhn-Munkres Algorithm: An Optimization Triaging Approach", *arXiv preprint arXiv:2202.01713*, 2022.

[30] A. Zeldes, "Can we Fix the Scope for Coreference? Problems and Solutions for Benchmarks beyond OntoNotes", *arXiv preprint arXiv:2112.09742*, 2021.

[31] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification", *Adv Neural Inf Process Syst*, Vol. 28, 2015.