Automatic Open Domain Information Extraction from Indonesian Text

Yohanes Gultom
Faculty of Computer Science
University of Indonesia
Email: yohanes.gultom@ui.ac.id

Wahyu Catur Wibowo
Faculty of Computer Science
University of Indonesia
Email: wibowo@cs.ui.ac.id

Abstract—**Availability of big amount digital documents calls for an automatic method to extract information from any text document regardless of domain. Unfortunately, existing open domain information extraction (open IE) systems are not suitable for low-resource language such as Indonesian. This paper introduces a system to extract relation triples from Indonesian text using rule-based triple candidates generator, rule-based token expander and machine-learning-based triple selector. Trained using our 2,344 triples dataset (166 positives & 2,183 negatives), a Random Forest triple selector model achieves cross-validation score of 0.58 F1 (0.62 precision and 0.58 recall).**

1. Introduction

Open domain information extraction (open IE) is a paradigm that facilitates domain-independent discovery of triple relations from text document [1]. It extracts relations from sentence in three-values tuples or triples format (x, r, y) where x and y called arguments and r is the relation [2]. In more linguistic term, the arguments are also referred as subject and object while relation are referred as predicate [3]. The example of this extraction is described in Figure 1.

As described in Table 1, unlike traditional information extraction (IE), open IE extracts domain-independent relations from sentence. While it retrieves relations in format of triples similar to knowledge extraction (KE), open IE doesn't follow whole Resource Data Format (RDF) specification[1] like KE [4] [5]. Although mapping to existing relation schema is required in real word task such as slot filling [3], ontology is not in the scope of open IE research. Open IE has also been reported to be useful for tasks such as question answering [6] and information retrieval [7].

Due to the nature of NLP tasks and heuristics used in open IE system, it is only applicable for a specific language [1]. So in order to extract open domain information from Indonesian text, a specific system has to be defined for this language. Furthermore, considering the scarcity of Indonesian NLP resources, the system need to effectively utilize them to achieve the objective. Through this paper, we propose a open IE system that addresses these issues.

1. https://www.w3.org/RDF/

TABLE 1. GENERAL COMPARISON BETWEEN TRADITIONAL INFORMATION, OPEN DOMAIN INFORMATION AND KNOWLEDGE EXTRACTION

	IE	**Open IE**	**KE**
Domain	Closed	Open	Open
Format	Depends on domain	Triples	RDF Triples
Ontology	Not available	Optional	Mandatory

Input
"Sembungan adalah sebuah desa yang terletak di kecamatan Kejajar, kabupaten Wonosobo, Jawa Tengah, Indonesia."

Output
1. (Sembungan, adalah, desa)
2. (Sembungan, terletak di, kecamatan Kejajar)

Figure 1. Example of expected input and output of open domain information extraction

We propose an Open IE system that combine heuristics (rule-based) models and a supervised learning model to extract relation triples from Indonesian text. This approach only requires single manually annotated dataset which is required to train triple selector/classifier.

In general, the contributions from this research are:

- Open domain information extraction system for Indonesian text
- Open-source implementation of the system in public repository[2]
- Dataset of manually tagged triple candidates
- Reusable Indonesian NLP pipelines (lemmatizer, part of speech tagger, named-entity recognizer and dependency parser) built by extending Stanford CoreNLP[3] API

2. https://github.com/yohanesgultom/id-openie
3. https://stanfordnlp.github.io/CoreNLP

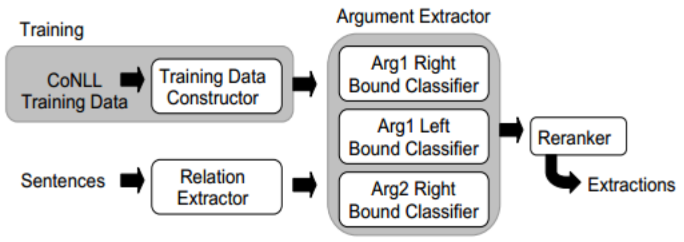

Training
CoNLL Training Data
Training Data Constructor
Argument Extractor
Arg1 Right Bound Classifier
Arg1 Left Bound Classifier
Arg2 Right Bound Classifier
Reranker
Extractions
Sentences
Relation Extractor

Figure 2. ArgLearner architecture training and extraction architecture

Further in this paper we will described some of the pre-eminent related works in open IE, the details about proposed system, experiments using some supervised-learning models as triples selector, analysis of the experiments results, and finally, conclusions and future works of this research.

2. Related Work

There has been plenty of works done in the open IE research. Starting from the introduction of open IE along with its first fully-implemented system, TextRunner, which further succeeded by systems built on top of it: ReVerb, R2A2 and Ollie (all from the same research group). The most recent research introduces Stanford OpenIE[4] which is an implementation of novel open IE system that outperforms Ollie in TAC-KBP 2013 Slot Filling task [3].

TextRunner was designed for massive size of open-domain web documents by avoiding heavy linguistic tasks and used inverted index to store extraction result [1]. It generates its own dataset (self-supervised) by using part of speech and dependency features and train a naive bayes classifier to select the triples. It argues that heavy linguistic tasks such as dependency parsing are not scalable to handle million of web documents. Additionally, it also uses redundancy assessor to remove redundant words (stop words, adverbs .etc).

ReVerb is an immediate successor of TextRunner which solves two significant problems in its predecessor: incoherent extractions and uninformative extractions [6]. It is composed of two algorithm: (1) Relation Extraction that extracts relations using syntactical and lexical constraint to solve the problems, and (2) Argument Extraction which retrieve the noun phrases as arguments of the relation. ReVerb takes as input a POS-tagged and NP-chunked and returns a set of relation triples.

R2A2 is a system built to fix argument extraction problem in ReVerb [2]. Instead of using heuristics to extract the arguments, it uses a learning-based system, ArgLearner, that accepts relation and sentence as inputs and returns the first (Arg1) and second arguments (Arg2). ArgLearner extracts the arguments using three classifiers based on REPTree and sequence labeling CRF as described in Figure 2.

Furthermore, Ollie (Open Language Learning for Information Extraction) utilizes ReVerb to learn open pattern

4. https://nlp.stanford.edu/software/openie.html

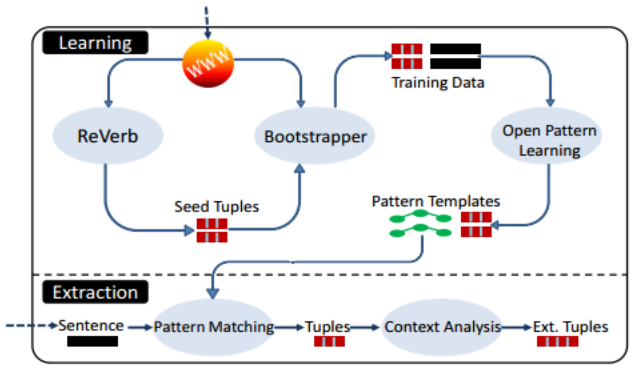


Figure 3. Ollie training and extraction architecture

templates to guide triples extraction from sentence. Additionally, Ollie does a context analysis to extend the tuples with contextual information in order to improve precision. Its training and extraction architecture is describe in Figure 3.

One of the most research proposes new open IE system that replaces the usage of large open patterns in Ollie with a set of fewer patterns for canonically structured sentences and a classifier that learns to extract self-contained clauses from a sentence [3]. This system is implemented in Stanford OpenIE which is also integrated in the populer open source suites, Stanford Core NLP.

3. Proposed System

As shown in the flowchart Figure 4, our system has three main components:

3.1. NLP Pipeline

The NLP pipeline is a series of NLP tasks that annotates one or more sentences and saves them in CONLL-U[5] format, a token-based sentence annotation format containing lemma, POS tag, dependency relation and a slot for additional annotation. This pipeline is buit by extending Stanford Core NLP classes and package them as single Java program (JAR). Those NLP tasks are:

1) **Tokenizer**
   We use default tokenizer provided by Stanford Core NLP, `PTBTokenizer` [8], which mimics Penn Treebank 3 tokenizer[6]. While this tokenizer provides many options to modify its behavior, we stick to default configuration that split sentence by whitelines to get the tokens.

2) **Part of Speech Tagger**
   We trained default Stanford Core NLP `MaxentTagger` [9] with Indonesian universal

5. http://universaldependencies.org/format.html
6. https://catalog.ldc.upenn.edu/LDC99T42

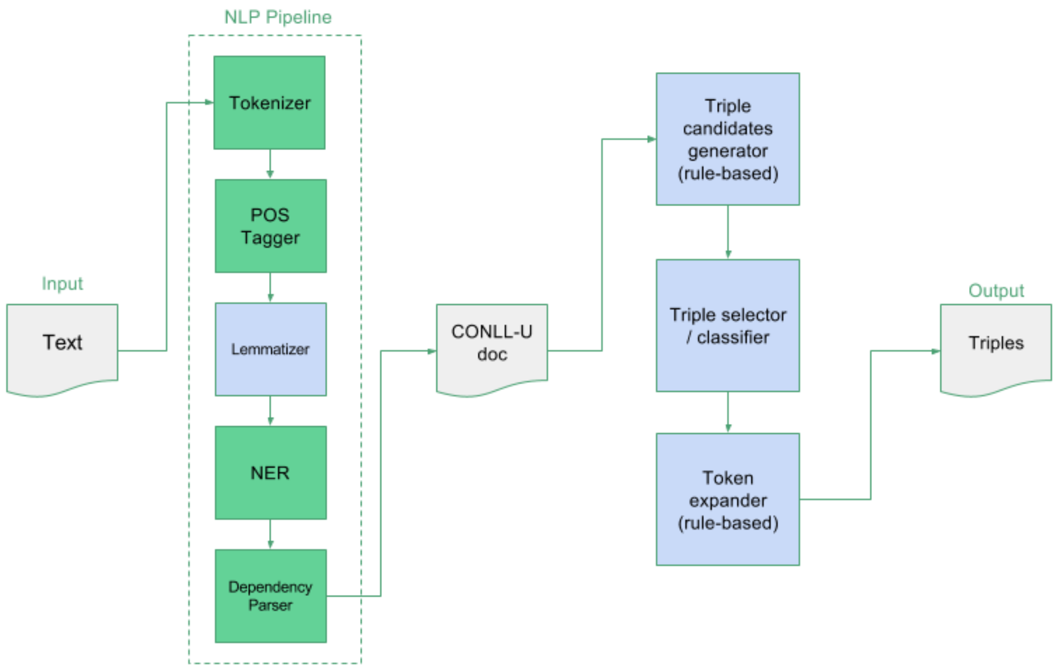

NLP Pipeline
Input
Text
Tokenizer
POS Tagger
Lemmatizer
NER
Dependency Parser
CONLL-U doc
Triple candidates generator (rule-based)
Triple selector / classifier
Token expander (rule-based)
Output
Triples

Figure 4. Indonesian open IE system flowchart

POS tag dataset which we convert from dependency parsing dataset[7]. This POS tagger uses Max Entropy (multi-class logistic regression) classifier which yields **93.68%** token accuracy and **63.91%** sentence accuracy when trained using 5,036 sentences and tested with 559 sentences from the dataset.

3) **Lemmatizer**
The lemmatizer used in this pipeline, `IndonesianLemmaAnnotator`, is implemented based on an existing Indonesian rule-based Lemmatizer [10] with some improvements:

- Reimplementation in Java language
- Usage of in-memory database to speed up dictionary lookup
- Integration with Stanford Core NLP annotator API for reusability

This lemmatizer yields **99%** accuracy when tested using dataset of 5,638 token-lemma pairs[8].

4) **Named-Entity Recognizer**

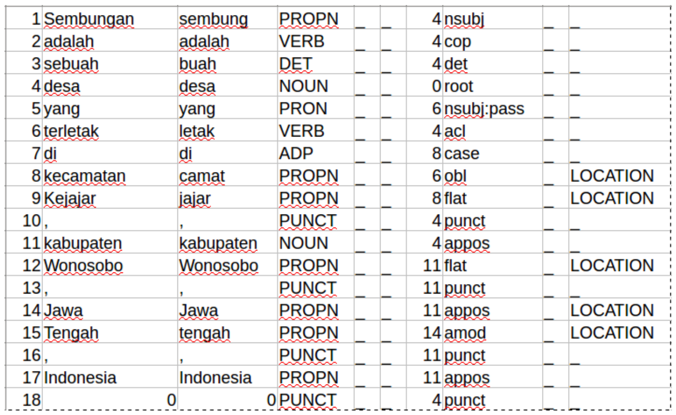
Stanford NLP `CRFClassifier` [11], a linear chain Conditional Random Field (CRF) sequence models, is trained using a dataset containing 3,535 Indonesian sentences with 5 entity class: Person, Organization, Location, Quantity and Time. When tested using 426 sentences, this models achieves 0.86 precision, 0.85 recall and **0.86** F1-score. The dataset itself is a combination between dataset from Faculty of Computer Science, University of Indonesia and a public dataset[9].

5) **Dependency Parser**

3.2. Triple Candidate Generator

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

7. https://github.com/UniversalDependencies/UD_Indonesian
8. https://github.com/davidchristiandy/lemmatizer
9. https://github.com/yusufsyaifudin/indonesia-ner



1	Sembungan	sembung	PROPN	_	_	4	nsubj	_	_
2	adalah	adalah	VERB	_	_	4	cop	_	_
3	sebuah	buah	DET	_	_	4	det	_	_
4	desa	desa	NOUN	_	_	0	root	_	_
5	yang	yang	PRON	_	_	6	nsubj:pass	_	_
6	terletak	letak	VERB	_	_	4	acl	_	_
7	di	di	ADP	_	_	8	case	_	_
8	kecamatan	camat	PROPN	_	_	6	obl	_	LOCATION
9	Kejajar	jajar	PROPN	_	_	8	flat	_	LOCATION
10	,	,	PUNCT	_	_	4	punct	_	_
11	kabupaten	kabupaten	NOUN	_	_	4	appos	_	_
12	Wonosobo	Wonosobo	PROPN	_	_	11	flat	_	LOCATION
13	,	,	PUNCT	_	_	11	punct	_	_
14	Jawa	Jawa	PROPN	_	_	11	appos	_	LOCATION
15	Tengah	tengah	PROPN	_	_	14	amod	_	LOCATION
16	,	,	PUNCT	_	_	11	punct	_	_
17	Indonesia	Indonesia	PROPN	_	_	11	appos	_	_
18	0	0	PUNCT	_	_	4	punct	_	_

Figure 5. CONLL-U Format Example

TABLE 2. TRIPLE CANDIDATE GENERATION RULES

Type	**Condition**
Subject	Token's POS tag is either PROPN, NOUN, PRON or VERB
	Token is not ”yang” nor ”adalah”
	Token's dependency is neither ”compound” nor ”name”
	Token's dependency is either ”compound” or ”name” but separated by more than 2 tokens from its head
Predicate	Token's position is after Subject
	Token's POS tag is either VERB or AUX
Object	Token's position is after Subject and Predicate
	Token's POS tag is either PROPN, NOUN, PRON or VERB
	Token is not ”yang” nor ”adalah”
	Token's dependency is neither ”compound” nor ”name”
	Token's dependency is either ”compound” or ”name” but separated by more than 2 tokens from its head

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

3.3. Triple Selector

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

3.4. Token Expander

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

TABLE 3. Triple selector features

#	Features
1	Subject token's POS tag
2	Subject token's dependency relation
3	Subject token's head POS tag
4	Subject token's named entity
5	Subject token's distance from predicate
7	Subject token's dependency with predicate
8	Predicate token's POS tag
9	Predicate token's dependency relation
10	Predicate token's head POS tag
11	Predicate token's dependents count
12	Object token's POS tag
13	Object token's dependency relation
14	Object token's head POS tag
15	Object token's named entity
16	Object token's dependents count
17	Object token's distance from predicate
18	Object token's dependency with predicate

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

4. Experiments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem.

TABLE 4. Token expansion rules for Subject or Object token

#	Condition for Subject or Object Token	Action
1	If dependent's relation to the token is either compound, name or amod	Expand
2	If dependent has same named entity as the token	Expand
3	If dependent and the token are wrapped by quotes or double quotes	Expand
4	If the head is a sentence root	Ignore
5	If dependent's POS tag is CONJ or its form is either , (comma) or / (slash)	Ignore
6	If dependent's POS tag is either VERB or ADP	Ignore
7	If dependent has at least one dependent with ADP POS tag	Ignore
8	If the first or last token in expansion result has CONJ or ADP POS tag	Remove
9	If the first or last index of expansion result is an incomplete parentheses symbol	Remove
10	If the last index of expansion result is yang	Remove
11	Else	Ignore

TABLE 5. Token expansion rules for Predicate token

#	Condition for Predicate Token	Action
1	If dependent is tidak	Expand
2	Else	Ignore

TABLE 6. Triple selector models performance

Models	P	R	F_1
Logistic Regression	0.64	0.28	0.36
SVM	**0.68**	0.41	0.51
MLP	0.54	0.46	0.47
Random Forest	0.62	**0.58**	**0.58**

Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

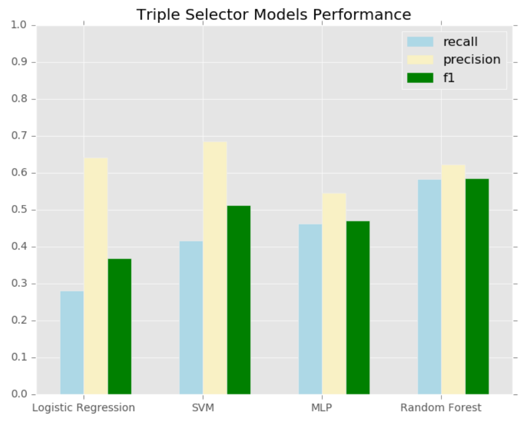


Figure 6. Triple selector models performance comparison chart

5. Analysis

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

6. Conclusion

Future works Extract implicit relations in sentence using better heuristics Use machine learning model for token expansion Estimate confidence level in every phases (NLP pipelines, candidate generator, triple selector, token expander) for following process Use better feature extraction using Word2Vec and deep learning Optimize system to handle large number of documents

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

References

[1] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web." in *IJCAI*, vol. 7, 2007, pp. 2670–2676.

[2] O. Etzioni, A. Fader, J. Christensen, S. Soderland *et al.*, "Open information extraction: The second generation," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[3] G. Angeli, M. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," *The semantic web*, pp. 722–735, 2007.

[5] P. Exner and P. Nugues, "Refractive: An open source tool to extract knowledge from syntactic and semantic relations." in *LREC*, 2014, pp. 2584–2589.

[6] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 1535–1545.

[7] O. Etzioni, "Search needs a shake-up," *Nature*, vol. 476, no. 7358, pp. 25–26, 2011.

[8] C. Manning, T. Grow, T. Grenager, J. Finkel, and J. Bauer, "Ptbtokenizer."

[9] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 173–180.

[10] D. Suhartono, "Lemmatization technique in bahasa: Indonesian," *Journal of Software*, vol. 9, no. 5, p. 1203, 2014.

[11] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 363–370.