Knowing Right from Wrong: Should We Use More Complex Models for Automatic Short-Answer Scoring in Bahasa Indonesia?

Ali Akbar Septiandri[1], **Yosef Ardhito Winatmoko**[2], **Ilham Firdausi Putra**[3]
[1]Universitas Al Azhar Indonesia, Jakarta, Indonesia
[2]Jheronimus Academy of Data Science, 's-Hertogenbosch, The Netherlands
[3]Institut Teknologi Bandung, Bandung, Indonesia
aliakbar@if.uai.ac.id, yosefardhitowin@gmail.com
ilhamfputra31@gmail.com

Abstract

We compare three solutions to UKARA 1.0 challenge on automated short-answer scoring: single classical, ensemble classical, and deep learning. The task is to classify given answers to two questions, whether they are right or wrong. While recent development shows increasing model complexity to push the benchmark performances, they tend to be resource-demanding with mundane improvement. For the UKARA task, we found that bag-of-words and classical machine learning approaches can compete with ensemble models and Bi-LSTM model with pre-trained word2vec embedding from 200 million words. In this case, the single classical machine learning achieved less than 2% difference in F1 compared to the deep learning approach with $\frac{1}{18}$ time for model training.

1 Introduction

Automated short-answer scoring is the application of computer technologies to assist human grader in evaluating the score of written answers (Dikli, 2006). The first track of UKARA 1.0 is the binary classification version of short-answer scoring, where participants are expected to develop a model that can distinguish right and wrong answers in free text format. The organizer published the questions, the labels' responses, and the guideline on how to determine whether an answer is acceptable.

During a period of five weeks, the training set and the development set were available, and we could validate our model through the score of the development set in a leaderboard. Subsequently, the test set was released, which consists of roughly four times the size of the development set. We are required to submit predicted labels based on the model that we have developed, and the winner was determined by the F1 score of the submitted prediction.

Our final submission to this task consists of feature extraction, such as n-grams and TF-IDF, and classical machine learning algorithms, namely logistic regression and random forest. Moreover, we tested a combination of classical algorithms through ensemble learning and deep learning approach. We have published our implementation for reproduction[1]. We discuss the dataset and our approach in the following sections.

2 Datasets

The dataset consists of two questions and the respective responses collected by the organizer of the challenge. All questions and responses are in Indonesian. The first question ("Task A") asked about the consequence of climate change. Concretely, what are the potential problems faced by a climate refugee when they migrate to a new place? The second question, referred to as task B, is based on an experiment. Potential customers initially wanted to buy clothes, prefer to donate the money instead, when they are presented with videos of the clothes manufacturing worker condition before paying. The respondents were required to give their opinion on why do people decided to change their minds. The responses statistics for both tasks are shown in Table 1.

	Task A	**Task B**
#Right Ans. Train	191(71%)	168(55%)
#Wrong Ans. Train	77(29%)	137(45%)
Avg. #Char	87.23	97.33
#Dev	215	244
#Test	855	974

Table 1: Summary statistics of the dataset.

[1]https://github.com/aliakbars/ukara

3 Methodology

We split our approach into three categories. Firstly, we have a single classical approach, where we employed simple logistic regression and random forest. Next, we experimented with ensemble learning by combining four different classical machine learning algorithms. Finally, an LSTM-based neural network model is applied. We elaborate on the preprocessing and modeling steps in the following paragraphs.

3.1 Preprocessing

For the preprocessing steps, we first tokenized and lemmatized the text using Bahasa Indonesia tokenizer provided by spaCy (Honnibal and Montani, 2017). We then extracted the features using bag-of-words or TF-IDF. Since the resulting matrix from this feature extraction method tends to be sparse and to encode token relations, we applied Latent Semantic Analysis (LSA) using Singular Value Decomposition (SVD) (Deerwester et al., 1990) on the matrix.

Based on our observation, we noticed that the labels of the provided training set are highly inconsistent. Some responses are clearly labeled incorrectly. For illustration, in task A we found ”*untuk pindah ke daerah yang aman*” (to move to a safe place) labelled as 1 (correct) while clearly it does not fit the criteria based on the guideline. The mislabeling was even more prominent in task B: ”*karana dengan menyumbang kita bisa membuat produksi pakaian menjadi lebih beretika*” (By donating, we can make clothes production becomes more ethical) is considered wrong while ”*agar upaya untuk membuat produksi pakaian menjadi lebih beretika.*” (As an effort to make clothes production becomes more ethical) is approved. To alleviate this issue, we decided to prepare a separate training set with manually corrected labels based on our own judgment. The correction result is shown in Table 2.

Finally, as the responses contain a lot of misspelled and slang words, we also experimented with simple spelling corrector using python difflib package and Indonesian colloquial dictionary (Salsabila et al., 2018). We tried every possible combination of preprocessing steps and whether to use an altered version of the training set with a parameter optimization library described in the following subsection.

	Original Label	**Corrected Label**	**Count**
Task A	wrong	right	10
	right	wrong	4
Task B	wrong	right	46
	right	wrong	13

Table 2: Corrected labels of the training set

3.2 Single Classical

After trying several machine learning algorithms, such as k-Nearest Neighbors, Naïve Bayes, logistic regression, and random forest, we found that random forest was the best model for task A. This corroborates what was found by Fernández-Delgado et al. (2014) in their comprehensive comparisons among several machine learning algorithms on different datasets. On the other hand, logistic regression with L2 regularization was the best for task B. The machine learning library used in this study is scikit-learn (Pedregosa et al., 2011). Since the dataset is quite small, we used 10-fold cross validation on the training set to avoid overfitting.

3.3 Ensemble Classical

In parallel, we experimented with ensemble model with a combination logistic regression, random forest, gradient boosting tree, and support vector machine. To find the best configuration for each model, we used hyperopt[2] library, which utilizes sequential model-based optimization (Bergstra et al., 2011). We trained separate voting-based ensemble models for task A and task B. The evaluation metric used for the optimization, including for the voting-ensemble model, is F1 score.

3.4 Deep Learning

Word embedding We pretrained Word2vec (Mikolov et al., 2013) 100 dimension word embedding using Gensim (Řehůřek and Sojka, 2010) on Indonesian text from Wikipedia dumps[3], Opensubs (Lison and Tiedemann, 2016), and the preprocessed UKARA dataset. For the word count details, see Table 3. The addition of text from Opensubs and UKARA dataset helps in providing informal words that are usually absent in Wikipedia articles. With the additional datasets, we ended up with a total of 420,024 unique vocabularies.

[2]http://hyperopt.github.io/hyperopt/
[3]https://dumps.wikimedia.org

Data Source	Word Count
Opensubs	105,348,108
Wikipedia	101,251,643
UKARA	36,930
Total	206,636,681

Table 3: Word counts for each data source

Modelling We used Keras (Chollet et al., 2015) with Tensorflow (Abadi et al., 2015) as the backend to build the model. The text was tokenized and padded into maximum length of 43 (90th percentile of all short-answer length) before it goes into the model. In order to build the embedding layer, we performed a multi-stage text processing using PySastrawi[4] stemmer and a normalizer function (removing duplicate adjacent characters) to minimize the amount of unknown vocabularies. For the known word counts found in each stage, see Table 4. This multi-stage process yields a total of 2.426 known vocabularies and 390 unknown vocabularies. We fit the model with EarlyStopping and ReduceLROnPlateau callbacks and Adam optimizer.

Experiment We ran the experiment on RepeatedStratifiedKFold with 10 split and 10 repeats. For each split and repeat, we predicted the validation and test set. We later normalized the result according to how many predictions made, essentially performing ensemble of 100 different models.

Stage	Known Word Count
1: Raw Word	2310
2: Stemmed	65
3: Normalized	48
4: Stemmed	3

Table 4: The count of known word found in each stage of building the word embedding layer

4 Results

Best Configuration For the single classical approach, we found that setting the `n_estimators` to 200 yielded the best result on the development set. The rest of the hyperparameters followed the default values from the sklearn implementation, including for the logistic regression model. For the ensemble, we used four different algorithms: logistic regression, random forest, XGBoost, and SVM. In this approach, we leave all the hyperparameters as optimized using hyperopt. We found that hard voting mechanism (weighing based on the binary class) provides better results compared to soft mechanism (weighted average of the predicted probability). Finally, for the deep learning approach, we altered the probability threshold for Task B to 0.48 as they gave better F1 for the development set.

Different preprocessing methods resulted in different performances in the two tasks. Therefore, we varied the use of unigram or TF-IDF, and whether we should apply SVD to the resulting matrix. On the other hand, we found that it is always better to use the lemmatizer built on top of spaCy in this task. Moreover, removing stopwords did not contribute much to the performance on the training set.

Performance The local CV results can be seen in Table 5 and Table 6. As the primary metric of this challenge is the F1 score, it is clear from Table 5 that we should employ the TF-IDF weighting with random forest algorithm for task A. From what we can see in Table 6, TF-IDF + SVD with logistic regression works better for task B. Table 7 shows the performance of our best single models compared to the ensemble approach. We used the optimized ensemble model trained on the original and also the label-corrected training set (Ens+Upd).

Training Time To quantify the required resources of each model, we include the total training time of each method in Table 8. We conducted the training for all models in the following specification: 2-core Intel(R) Xeon(R) CPU @ 2.20GHz and 16GB RAM. As expected, the deep learning method demanded the longest time, almost 6 hours for training word2vec, task A, and task B. Meanwhile, we needed less than 20 minutes to find the best performing single classical models for both tasks.

5 Discussion

Based on the results shown in Table 7, all models perform almost equally well for task A. For task B, the label correction in the Ens+Upd model gives a definite boost on the training set F1 score. This improvement indicates conflicting labeling for task B. However, the difference for the uncorrected development set is just 0.003: even with the corrected labels, the model still perform similarly for the

[4]`https://github.com/har07/PySastrawi`

	Precision	**Recall**	**F1**
1-gram+RF	0.830 ± 0.082	0.916 ± 0.057	0.870 ± 0.063
1-gram+logreg	0.850 ± 0.093	0.890 ± 0.084	0.868 ± 0.081
1-gram+SVD+RF	0.794 ± 0.030	$\mathbf{0.984 \pm 0.025}$	0.879 ± 0.021
1-gram+SVD+logreg	$\mathbf{0.858 \pm 0.080}$	0.884 ± 0.074	0.869 ± 0.065
TF-IDF+RF	0.833 ± 0.066	0.963 ± 0.036	$0.892 \pm 0.045^{**}$
TF-IDF+logreg	0.743 ± 0.040	0.979 ± 0.037	0.844 ± 0.035
TF-IDF+SVD+RF	0.778 ± 0.030	$\mathbf{0.984 \pm 0.025}$	0.869 ± 0.020
TF-IDF+SVD+logreg	0.746 ± 0.040	0.979 ± 0.037	0.846 ± 0.036
word2vec+BiLSTM	0.856 ± 0.063	0.934 ± 0.045	$\mathbf{0.900 \pm 0.034}^{*}$

Table 5: 10-fold cross validation results from task A

	Precision	**Recall**	**F1**
1-gram+RF	0.699 ± 0.081	0.649 ± 0.123	0.667 ± 0.086
1-gram+logreg	$\mathbf{0.724 \pm 0.072}$	0.732 ± 0.135	0.723 ± 0.087
1-gram+SVD+RF	0.655 ± 0.077	0.768 ± 0.094	0.703 ± 0.065
1-gram+SVD+logreg	0.706 ± 0.055	0.714 ± 0.117	0.707 ± 0.074
TF-IDF+RF	0.693 ± 0.059	0.697 ± 0.125	0.691 ± 0.084
TF-IDF+logreg	0.708 ± 0.082	0.845 ± 0.118	0.767 ± 0.080
TF-IDF+SVD+RF	0.671 ± 0.077	0.850 ± 0.100	0.744 ± 0.047
TF-IDF+SVD+logreg	0.715 ± 0.083	0.839 ± 0.110	$0.768 \pm 0.075^{**}$
word2vec+BiLSTM	0.705 ± 0.077	$\mathbf{0.884 \pm 0.086}$	$\mathbf{0.778 \pm 0.048}^{*}$

Table 6: 10-fold cross validation results from task B

	Train A	**Train B**	**Dev**	**Test**
Single	0.892	0.768	0.793	0.800
Ens+Ori	0.885	0.764	0.799	0.802
Ens+Upd	0.898	**0.831**	0.802	0.804
Deep learning	**0.900**	0.772	**0.806**	**0.811**

Table 7: F1 score comparison with the alternative ensemble models

	Word2Vec	**Task A**	**Task B**	**Total**
Single	–	**8.98**	**10.07**	**19.05**
Ens+Ori	–	36.53	36.70	73.23
Ens+Upd	–	44.45	45.78	90.23
Deep Learning	79.15	132.42	132.06	343.63

Table 8: Total training time for each method (in minutes)

holdout datasets. Since we did not observe significant improvement, we used only original labels to train the deep learning model.

To analyze how hard it is to separate the right from the wrong answers, we reduced the dimensionality of the data into 2D using 1-gram, SVD, and t-SNE (Maaten and Hinton, 2008). Figure 1 suggests that it is harder to separate the two answers in task B. On the other hand, we see more concentrated data points from wrong answers in task A. A similar phenomenon can also be observed in Figure 2 and Figure 3. We can see a lot more data points in the 0.4-0.6 prediction range in Figure 3.

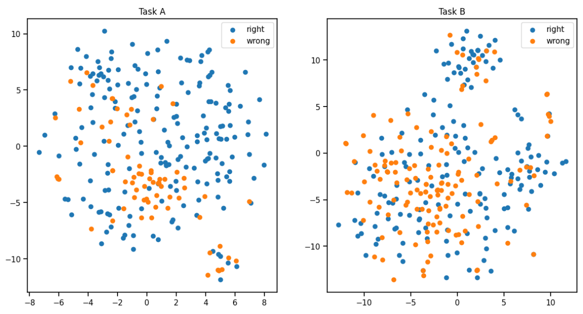

Figure 1: t-SNE visualisation

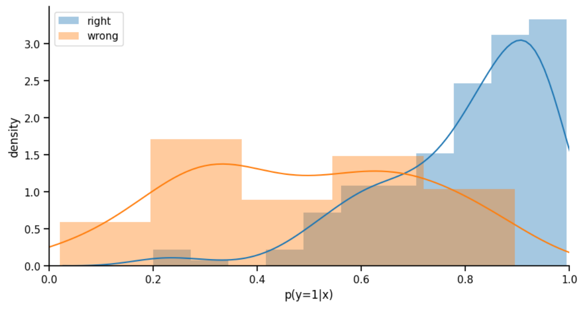

Figure 2: Single model prediction (random forest) with probability on Task A

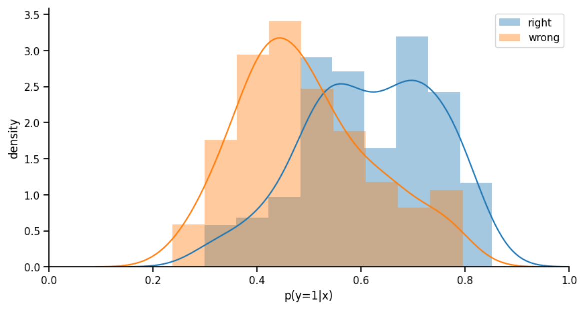

Figure 3: Single model prediction (logistic regression) with probability on Task B

6 Related Work

Studies of short-answer scoring methods range from using pattern matching (Mitchell et al., 2002; Leacock and Chodorow, 2003; Sukkarieh et al., 2004; Nielsen et al., 2009), semantic similarity (Mohler and Mihalcea, 2009; Mohler et al., 2011; Heilman and Madnani, 2013; Jimenez et al., 2013), to neural architectures (Riordan et al., 2017). Note that short-answer scoring tasks usually have shorter response length compared to essay scoring tasks (Riordan et al., 2017). Short-answer scoring focuses on content only instead of broader writing quality, such as elaboration, organization, and grammar (Burstein et al., 2013).

In semantic similarity, Mohler and Mihalcea (2009) compared TF-IDF, WordNet, and LSA-based models for short-answer scoring. Mohler et al. (2011) added graph alignment scores to add syntactic knowledge to improve the LSA-based model in the prior study. Sultan et al. (2016) and Baroni et al. (2014) introduced the use of word embedding using word2vec for this task. Our word2vec model is pretrained using a large collection of Indonesian text from Wikipedia dumps, Opensubs, and the preprocessed UKARA dataset. This makes our word embedding model different from what can be found in the previous work by Riordan et al. (2017).

In this study, we did not use WordNet due to the unavailability of a rich lexical similarity database in bahasa Indonesia. Moreover, in our shared task, the organiser also gave the binary labels for each answer and coding guidelines instead of the expected right answers.

7 Conclusions

In this report, we compare three approaches for automatic Indonesian short-answer scoring using the UKARA 1.0 dataset: single, ensemble, and deep learning. Albeit being more sophisticated, we found that the F1 score difference is insignificant when we compare the single model to the ensemble and the deep learning approach. The 1.1% increment in the deep learning model can help clinch the top position in a competition, but it is not worth the required resources for practical settings. We posit that the primary reason is due to the noisy labels. As the classical models require only $\frac{1}{18}$ of the total time to train deep learning models, we should use simple models and allocate more time for improving dataset quality.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

Jill Burstein, Joel Tetreault, and Nitin Madnani. 2013. The e-rater automated essay scoring system. *Handbook of automated essay evaluation: Current applications and new directions*, pages 55–67.

François Chollet et al. 2015. Keras. `https://keras.io`.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Semire Dikli. 2006. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, 5(1).

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181.

Michael Heilman and Nitin Madnani. 2013. Ets: Domain adaptation and stacking for short answer scoring. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 275–279.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Sergio Jimenez, Claudia Becerra, and Alexander Gelbukh. 2013. Softcardinality: Hierarchical text overlap for student response analysis. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 280–284.

Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):389–405.

P. Lison and J. Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards robust computerised marking of free-text responses. In *Proceedings of the 6th CAA Conference*. Loughborough University.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 752–762.

Michael Mohler and Rada Mihalcea. 2009. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 567–575. Association for Computational Linguistics.

Rodney D Nielsen, Wayne Ward, and James H Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(4):479–501.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chungmin Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168.

Nikmatun Aliyah Salsabila, Yosef Ardhito Winatmoko, Ali Akbar Septiandri, and Ade Jamal. 2018. Colloquial indonesian lexicon. In *2018 International Conference on Asian Language Processing (IALP)*, pages 226–229. IEEE.

Jana Z Sukkarieh, Stephen G Pulman, and Nicholas Raikes. 2004. Auto-marking 2: An update on the ucles-oxford university research into using computational linguistics to score short, free text responses. *International Association of Educational Assessment, Philadephia*.

Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. 2016. Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075.