Shared Task in Evaluating Accuracy: Leveraging Pre-Annotations in the Validation Process

Nicolas Garneau and Luc Lamontagne
Université Laval, Québec, Canada
{nicolas.garneau,luc.lamontagne}@ift.ulaval.ca

Abstract

We hereby present our submission to the Shared Task in Evaluating Accuracy (Reiter and Thomson, 2020) at the INLG 2021 Conference. Our evaluation protocol relies on three main components; rules and text classifiers that pre-annotate the dataset, a human annotator that validates the pre-annotations, and a web interface that facilitates this validation. Our submission consists in fact of two submissions; we first analyze solely the performance of the rules and classifiers (pre-annotations), and then the human evaluation aided by the former pre-annotations using the web interface (hybrid). The code for the web interface and the classifiers is publicly available[1].

1 Introduction

Evaluating Data-to-Text natural language generation (NLG) systems is a very important task despite its notorious difficulty (Thomson and Reiter, 2020). Reiter and Thomson introduced a Shared Task in Evaluating Accuracy which consists of factually assessing the accuracy of basketball games summaries produced by an automatic (neural) language generator. The underlying Data-to-Text dataset was originally created by Wiseman et al..

Evaluating the accuracy of a game's summary relies on identifying the errors within the generated text. Borrowing the same terminology as Reiter and Thomson, the set of possible errors comprises 6 different categories; *Number*, *Named entity*, *Word*, *Context*, *Not Checkable*, and *Other*. We refer the reader to the original paper for more details about these different error categories.

As part of the shared task, participants were asked to propose an automatic evaluation metric (or algorithm) and/or an evaluation methodology that could be followed by humans in order to assess the accuracy of a given generated text. In our case, we proposed both, i.e. a hybrid approach where the evaluation methodology leverage pre-annotations in order to accelerate (and hopefully improve) the evaluation process.

2 Evaluating Accuracy

Our evaluation methodology relies on the pre-annotation of game summaries and a validation procedure that we describe in the following sections.

2.1 Pre-Annotation of Game Summaries

In order to accelerate the task of evaluating the accuracy, we propose to pre-annotate the game summaries (i.e. identify *potential* errors) using a set of rules and text classifiers. To this end, we followed the hierarchical proposition specified by Reiter and Thomson[2] and designed our system accordingly.

We then separate the set of errors into two groups. The first group contains *Number* and *Name* errors, where every instance can easily be identified (not necessarily validated) by an algorithm. This includes names beginning with a capital letter and numbers consisting of either digit (1, 2, etc.), written words (one, two, etc.) or ordinals (first, second, etc.). The second group contains all other error types, i.e. *Word*, *Context*, and *Not Checkable* [3]. We illustrate the distribution of errors on the development set in Table 1

For the first group of errors, we designed two simple rules. Given the set of n *Number* instances $\{u_i | i \in 1 \dots n\}$ and m *Name* instances $\{a_j | j \in 1 \dots m\}$ in a given sentence, we check for the following;

[1] https://github.com/ngarneau/accuracySharedTask

[2] *Number* > *Name* > *Word* > *Context* > *Not Checkable* > *Other*

[3] Since the *Other* type of error does not provide many examples and it was designed for nonsensical text, we do not consider it in our submission.

Error Type	Count
Number	474
Name	317
Word	334
Context	51
Not Checkable	37

Table 1: Distribution of error types in the development set for the first group (*Number* and *Name*) and the second group (*Word*, *Context*, and *Not Checkable*).

1. For every pair (u_i, a_j), find a correspondence (row–column wise) in the Box Score. If the check fails, we consider it is a *Number* error.

2. For every a_j, find a correspondence (anywhere possible) in the Box Score. If the check fails, we assume it is a *Name* error.

These checks are done to pre-annotate sentences with *Number* and *Name* errors which are later validated by the annotators. We provide more details on the validation step in Section 2.2.

The errors of the second group are much more difficult to identify. Since the annotators might not be native English speakers or might have little knowledge about the basketball lexical field, we help them with two textual classifiers that are trained on the development set released along with the task. The first classifier predicts if a sentence may contain errors belonging to the second group or not (i.e. a binary classification). The second classifier predicts which type(s) of errors may be present within a sentence positively labeled to the second group. Hence it is a multiclass, multilabel classifier. It takes as input only the sentence, no contextual information from the box score. This classifier uses unigrams, TF-IDF weights (Sparck Jones, 1988) and a multinomial logistic regression model. The regression model assigns to each word w_i in the vocabulary a score specific to a class c_j, namely $s_{i,j}$. Using the TF-IDF weight t_i and a classification threshold τ, we classify a word as being erroneous w.r.t. a specific class c_j as follows:

$$c_{i,j} = \begin{cases} 1, & \text{if } t_i \cdot s_{i,j} \geq \tau \\ 0, & \text{otherwise} \end{cases}$$

$c_{i,j}$ is thus used as a pre-annotation to the word w_i. If multiple error labels can be associated with a word, we take the one with the highest importance. We used $\tau > 0.5$ in our experiments.

The binary and multilabel classifiers have been trained using the scikit-learn library (Pedregosa et al., 2011) and achieve respectively 0.63 and 0.76 F1-scores on the development set.

2.2 Evaluation Procedure

Our evaluation procedure is composed of three steps, where the first and the last are fully automated while the second is performed by a human.

Importing games data and pre-annotation. The annotator first imports the games' data into the validation database with a python script. Then, the game summaries are automatically pre-annotated using the classification models described in Section 2.1. The games are then ready to be validated manually.

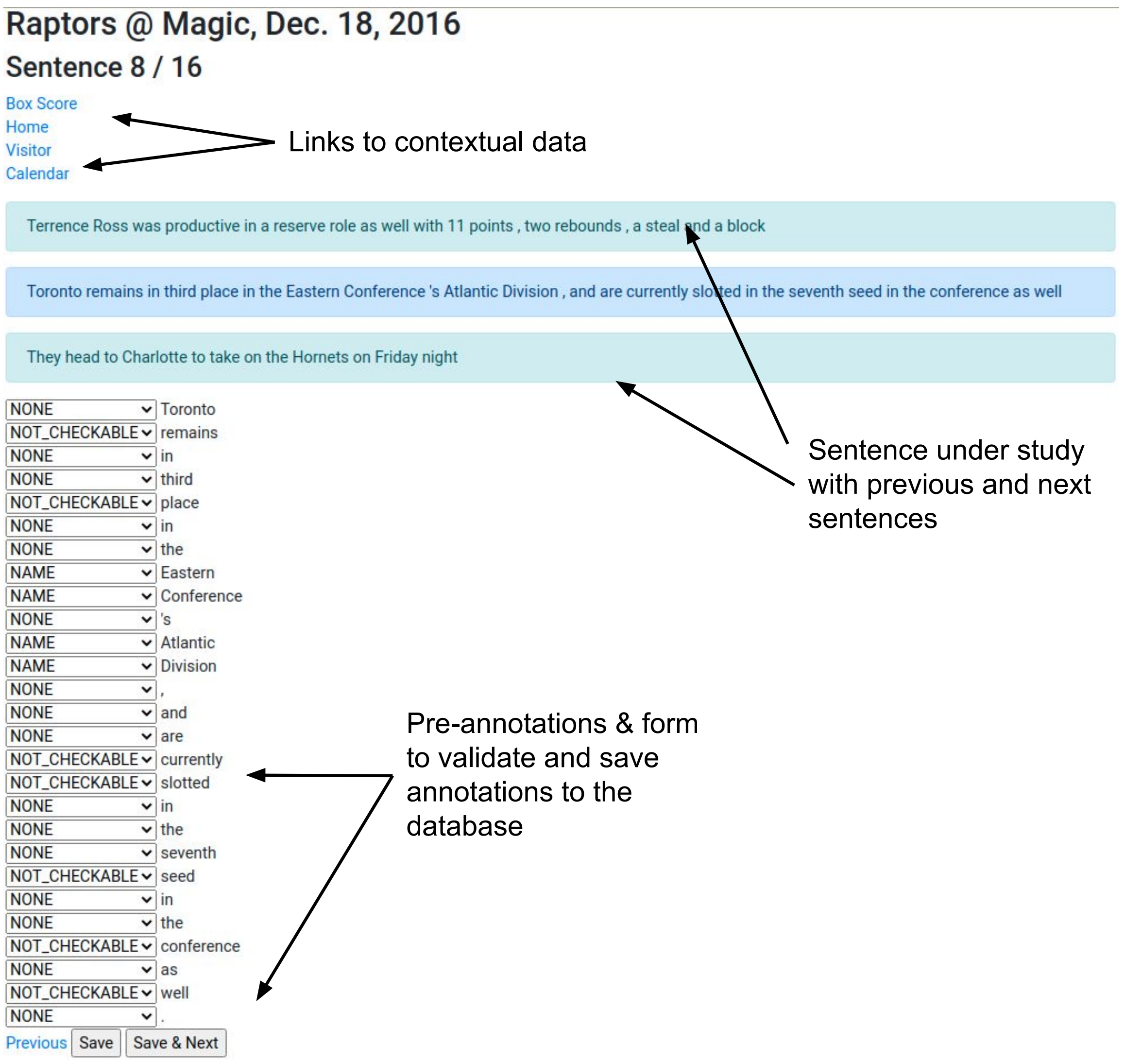
Validating pre-annotations. The list of games to validate is displayed through the web interface to the annotator. The annotator selects the game to annotate and begins with the first sentence. To provide a little more context, we include links to the box score, to each team's respective pages, and to the current calendar showing the dates from the month in which the game was played. We also present the previous and next sentences to the current one, if there are any. Then, a list of all the words of the sentence under study is presented, with a dropdown list filled with pre-annotations that lets the user select the possible error a given word might be associated with. The annotator is asked to follow the evaluation guidelines provided by Reiter and Thomson. Once the validation is done, the user can save them and move onto the next sentence until the summary is fully validated. We illustrate in Figure 1 the main interface proposed to the annotator.

Preparing submission file. Once all the games have been validated, the annotator needs to prepare a submission file. This file is created using a python script and made available to the evaluators.

3 Results

3.1 Pre-Annotations

We can see from Table 2 that the pre-annotations offer a basic coverage for the *Number* and *Name* errors. Recall that these pre-annotations rely strictly on two simple rules.


Links to contextual data
Sentence under study with previous and next sentences
Pre-annotations & form to validate and save annotations to the database

Raptors @ Magic, Dec. 18, 2016

Sentence 8 / 16

Box Score
Home
Visitor
Calendar

Terrence Ross was productive in a reserve role as well with 11 points , two rebounds , a steal and a block

Toronto remains in third place in the Eastern Conference 's Atlantic Division , and are currently slotted in the seventh seed in the conference as well

They head to Charlotte to take on the Hornets on Friday night

NONE	Toronto
NOT_CHECKABLE	remains
NONE	in
NONE	third
NOT_CHECKABLE	place
NONE	in
NONE	the
NAME	Eastern
NAME	Conference
NONE	's
NAME	Atlantic
NAME	Division
NONE	,
NONE	and
NONE	are
NOT_CHECKABLE	currently
NOT_CHECKABLE	slotted
NONE	in
NONE	the
NONE	seventh
NOT_CHECKABLE	seed
NONE	in
NONE	the
NOT_CHECKABLE	conference
NONE	as
NOT_CHECKABLE	well
NONE	.

Previous Save Save & Next

Figure 1: The main interface for the annotator used to validate the pre-annotations. Links to the box score, teams' statistics, and current calendar are available at the top. The previous and next sentences are shown to the annotator to provide more context. The annotator validates the pre-annotations by updating the dropdown lists in the form.

		Pre-Annotations						Human Validation					
	Error	**Num.**	**Name**	**Word**	**Cont.**	**N/C**	**Avg.**	**Num.**	**Name**	**Word**	**Cont.**	**N/C**	**Avg.**
DEV	**Precision**	0.38	0.67	0.20	0.00	0.10	0.31	0.65	0.78	0.33	0.08	0.16	0.49
	Recall	0.30	0.53	0.58	0.00	0.60	0.49	0.58	0.69	0.67	0.16	0.60	0.66
TEST	**Precision**	0.35	0.79	0.14	0.00	0.19	0.33	0.88	0.94	0.73	0.40	0.39	0.88
	Recall	0.44	0.59	0.36	0.00	0.50	0.50	0.86	0.92	0.68	0.75	0.24	0.84

Table 2: Precision and Recall results for every error types on the development and test set. We present the difference between the pre-annotations only, and the human validation.

The classifiers, however, struggle to precisely identify erroneous words (while having a decent recall). During validation of the summaries, we noticed that the classifiers acted more like pointers. Indeed, when there were potential *Word*/*Context*/*Not Checkable* errors, the classifiers flag non-erroneous words. Take for example the sentence in Figure 1. The classifier identified the first words like "remains" and "place" as being *Not Checkable* errors. While these words do not correspond to the exact error ("*third* place"), it does give a pointer to the annotator that there is something to verify within this particular phrase. Nonetheless, it did help the annotator to give more serious attention to these sentences, and especially to detect *Context* errors.

We can see from Table 2 that there are negligible differences across error types between the development and test set. This suggests that the pre-annotations did not overfit.

3.2 Human Evaluation

The human evaluation was conducted in two subsequent stages; the annotator first validated the development set and then the test set, and a difference in the results obtained for these two data sets can be observed in Table 2. This difference is attributable to the fact that the annotator gained domain knowledge throughout the annotation process. This is especially true for the *Word* and *Context* errors. Overall, on the test set, we achieve interesting (and surprisingly high) precision and recall scores of 0.88 and 0.84 respectively. It would have been interesting to validate the test set with an annotator that did not have any prior knowledge on the task and domain i.e. had not seen the development set.

3.3 Evaluation Time

Familiarisation with the problem, the dataset, and the domain took roughly 4 hours. The development of the evaluation interface, rule modeling, and creation of the classifiers represent an effort of one day (8 hours). The training time of the classifiers is near-instantaneous, and the computing power needed is CPU only. The evaluation time required for one game is between 10 to 15 minutes. Since we save the normalized annotations within the database, it is easy to generate the submission file with a simple script that takes seconds to run. Overall, considering the validation of both development and test sets, conducting the whole experiment took around 30 to 35 hours. Originally, the protocol took around 30 minutes per annotator and used 3 relatively knowledgeable annotators to achieved good results. We thus considerably reduced the manual effort.

4 Discussion

As previously mentioned, evaluating the accuracy of generated text in a Data-to-Text setting is a very hard task. In this experiment, the annotators found the task especially difficult since they are not native English speakers and basketball game descriptions are not (or were not!) a domain with which they were familiar.

While the following example may seem trivial, it took several summaries for the annotator to understand that “led the bench” means that it is targeting non-starting players (one could easily assume that every player starts on the bench). Also, an annotator who knows which player plays for which team will greatly improve and accelerate the evaluation process. This is something that the annotator just barely began to get comfortable with towards the end of the task.

Our experiments, due to the lack of time and resources, do not explicitly expose the benefits of having pre-annotations. It would have been interesting to see if, given two annotators with the same prior knowledge[4], the pre-annotations helps them to solve the task both in a matter of time and accuracy, hence evaluating the inter-annotator agreement. Nonetheless, the pre-annotations did help the annotators through the evaluation process in three different ways;

1. get familiar with the task, domain, and dataset by seeing which potential word may be erroneous;
2. identify errors that the annotator would have missed;
3. save time and effort.

We decided to design our own annotation tool instead of using WebAnno (Yimam et al., 2013), the one initially used by the authors of the task. This decision was mainly motivated by the flexibility of storing and pre-annotating the game summaries. In future works, we would consider more robust text classifiers coupled with active learning in order to improve the pre-annotations, while executing the task. We would also consider adding contextual information from the box score for the classification.

References

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.

[4]Then again, it is hard to measure *prior knowledge* on a specific domain.

Ehud Reiter and Craig Thomson. 2020. Shared task on evaluating accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 227–231, Dublin, Ireland. Association for Computational Linguistics.

Karen Sparck Jones. 1988. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.