Automatic Verification of Data Summaries

Rayhane Rezgui
EURECOM, France
Rayhane.Rezgui@eurecom.fr

Mohammed Saeed
EURECOM, France
Mohammed.Saeed@eurecom.fr

Paolo Papotti
EURECOM, France
Paolo.Papotti@eurecom.fr

Abstract

We present a generic method to compute the factual accuracy of a generated data summary with minimal user effort. We look at the problem as a fact-checking task to verify the numerical claims in the text. The verification algorithm assumes that the data used to generate the text is available. In this paper, we describe how the proposed solution has been used to identify incorrect claims about basketball textual summaries in the context of the Accuracy Shared Task at INLG 2021.

1 Introduction

Natural Language Generation (NLG) can be used to generate texts out of relational data, where the goal is to have correct and clear statements that describe what can be found in tables (Gong et al., 2019). However, this is not always the case, since NLG tools, although producing correct sentences grammar-wise, sometimes fail to generate accurate texts, eventually containing some factual errors (Wiseman et al., 2017).

The goal of our work is to evaluate generated texts by identifying numerical claims and fact-checking them with the relational data available at hand. We apply different techniques on the provided summaries and use the available relational data about the matches to state whether the claims are true or false. The idea behind fact checking using relational tables is to create an automated verification pipeline using data-driven algorithms, such as deep learning models (Nakov et al., 2021; Saeed and Papotti, 2021). Building upon previous work in fact-checking statistical claims (Karagiannis et al., 2020), we focus on such kind of claims due to the availability of trustworthy relational tables to verify them. The goal is not only to be accurate, but also to limit the user effort, such as labelling data or writing scripts, in the setup of the system.

> The Memphis Grizzlies (5-2) defeated the Phoenix Suns (3 - 2) Monday 102-91 at the Talking Stick Resort Arena in Phoenix. The Grizzlies had a strong first half where they out-scored the Suns 59-42. Marc Gasol scored 18 points, leading the Grizzlies. Isaiah Thomas added 15 points, he is averaging 19 points on the season so far.

Figure 1: Example of a generated textual summary from the basketball relational data.

Team	PTS	AST	..	TOV
Kings	99	22	..	21
Nets	107	20	..	9

(a) Sample of team statistics in a match.

Team	PTS	AST	..	FGA
Ben McLemore	11	2	..	9
Sergey Karasev	5	3	..	5
..	..	..	..	..
Kevin Garnett	10	2	..	10

(b) Sample of player statistics in a match.

Statistical claims form a significant part of the set of claims in the shared task (around 40%) as the tables mostly contain numerical facts, rather than textual. Consider samples of the relational tables of players and teams in Tables 2a and 2b, respectively. Sentence "Sacramento Kings scored 99 points." is verified by identifying the team name (key value "Kings") and column (label "PTS"). A comparison between the value in the identified cell and the value in the text (99) validates the claim. More complex claims, such as "Kings defeated Nets", require comparisons between two cells.

To verify the claims above, our solution consists of three main steps: (i) claim identification, (ii) identification of properties that construct a val-

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 271–275,
Aberdeen, Scotland, UK, 20-24 September 2021.

idation query over the data for every claim, and (iii) claim verification. In the following, we first describe our solution and then report some experimental results and possible directions for future work.

2 Method

We target claims that can be verified computationally by using operations over the cell values in the tables (relations). We follow an approach inspired from previous work on computational fact-checking for statistical claims (Karagiannis et al., 2020). We begin the process by extracting claims from the input sentences (Section 2.1) and then identify query properties (Section 2.2), which are used for building the query that verifies the claim (Section 2.3).

Figure 3 represents the architecture of the solution, where we input sentences and get a collection of properties, including the claims to verify and the elements of the data that are needed for this task. Every sentence can contain more than one named entity (were we focus only on players/teams) as well as several claims. We have to associate each claim and the resulting properties to the entity in question. By looking at the summaries in the task, it is possible to observe that many sentences fall into two categories:

- *Comparative sentences* where the text describes both teams and compares the scores from both sides, e.g., "The Sacramento Kings defeated the Brooklyn Nets 107 - 99." In this case, we assume an order in the sentence. The first claim to be extracted is the one we associate to the first entity. We enumerate all the numerical claims and assign them to the first or second entity based on first appearance. For our example, 107 is assigned to Kings and 99 to Nets. As for the "defeated" claim, we associate it to both Kings and Nets.

- *Look-Up sentences* where the text specifies information/statistics about one entity. In this case, we just define that entity as the row of interest. Otherwise, we associate our claims to the first player to appear.

After claim and properties are identified, we use the latter to create a query that fact checks the claim. The query returns a Boolean value in the final output and it is self-explanatory, i.e., it is a declarative specification easy to interpret as an explanation of the checking process. An example of a query that returns the number of **points (PTS)** of the team **Sacramento Kings** from an identified table **t** is

```
SELECT t.PTS FROM t WHERE
t.Team='Kings';
```

whose output is compared against the value of the extracted claim.

2.1 Claim Identification

Following the data-generation procedure in (Karagiannis et al., 2020), we wrote 9 templates to generate natural-language sentence starting from the provided tables. These scripts could be replaced with NLG algorithms in a fully automatic solution (Parikh et al., 2020). The generated data is used to fine-tune a **BertForTokenClassification** model (Devlin et al., 2019) to identify claims in an input sentence. More precisely, the input sentence is tokenized and a binary-classification layer is applied on every token to predict whether the token is part of the claim or not. As some claims occur together, we rely on textual separators (like commas) to separate them. The following sentence shows an example of a sentence with six claims underlined: "AJ Hammons had **5 rebounds**, **10 assists**, **6 turnovers**, **7 points**, **3 steals** and **1 block**."

The model is able to correctly identify claims on the synthetic test dataset. However, we also tried another simpler regex-based approach where we focus on key words like the columns (points, turnovers, etc.) and extract the neighboring words. Both approaches gave similar results on sentences sampled from the dataset of the shared task, with BERT failing to extract the correct claims for some sentences. This is mainly due to two reasons. First, neural models are frail w.r.t. small changes in textual input (Zhang et al., 2020). Second, when multiple claims occur sequentially, there is no clear way of separating them unless some form of separator is found (like a comma). A simple fix to this would be to include spaces in the tokenized input and learn which spaces are included in a claim and which are not. The regex solution is pragmatic in cases where annotated data is not available as it produces results comparable to the classifier solution.

The regular expressions for claim identification check for trigger words (such as 'assists', 'blocks', 'field goals', 'minutes', 'points','rebounds', 'steals', 'turnovers') and then identify the following string in the text. This string can be a numerical value in its

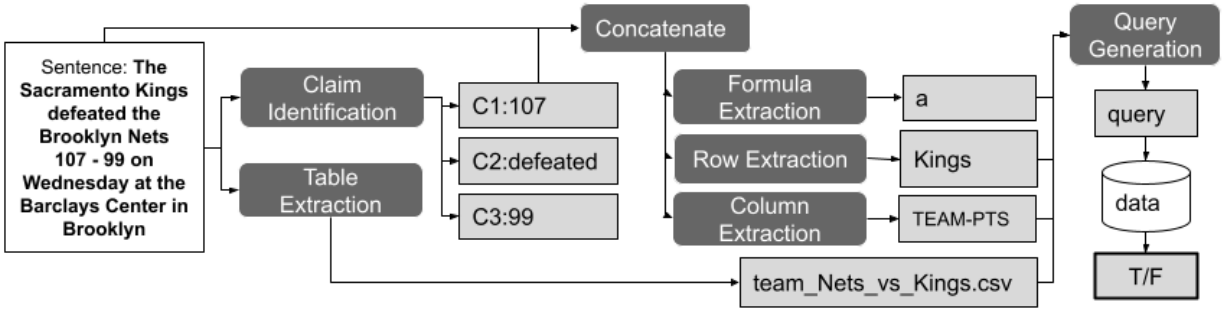

Figure 3: Architecture of the solution. Given a sentence, we identify the claims and the table to verify them. Then every claim (C1-C3 in the example) is processed individually to obtain a query for its verification. (We take claim C1 as an example in the figure.)

numerical format (15) or in words (fifteen). These expressions might identify some false positives. In the sentence "The only other Net to reach double figures in points was Ben McLemore", we identify a string ("in points") which cannot be verified as it does not contain a numerical value. We filter out such claims before verification. For comparison sentences, we consider "defeated", "outscored" and "lost" in the word list.

2.2 Property Identification

After identifying claims in the text summary, we ought to predict the data and the operation that allow us to verify them. We call *properties* the elements that ultimately enable the generation of a verification query for the given claim. Properties include: (i) the name of the table (relation), (ii) the primary key value (i.e., the identifier for a tuple in the relation), (iii) the column (i.e., the attribute label), as well as (iv) the formula, which include the simple look up of a value (a) and value comparisons (b>a, a>b). We describe the main modules, as depicted in Figure 3 next.

For the *column* and the *formula* extraction, we fine-tune a *BertForSequenceClassification* model (Devlin et al., 2019) with generated training data in Section 2.1. For the column identification, we have 15 possible classes, whereas for the formula identification we only have 3 classes with most examples pivoting on the "lookup class" (a). This is due to the fact that most relevant statistics are already reported in the tables, such as ratios (a/b), and they can be verified with a simple lookup.

For a given input text, extracting the team names is crucial as they enable identifying the *table* and primary *key value*(s) of a sentence. The given textual inputs describe a single basketball match which usually begin with a general sentence mentioning the associated teams. We therefore search in the first sentence of a given text for the mentioned team names. For every pair of teams, (i.e., for every match), we consider one table reporting the team statistics (Figure 2a) and another one reporting player statistics (Figure 2b). These are used later on for other sentences in the same summary, where team/player names are identified accordingly to the associated tables.

There are sentences where both team and player names occur. In this case, we use the player name as the primary key value, since claims are more likely to be related to the player than to the team. Out of 257 sentences that were extracted from the test files, 22 sentences contain more than 2 names, with at least one of them being a player name, such as "**Bradley Beal** led the way for the **Wizards** with a game - high 18 points , which he supplemented with five rebounds , four assists and one steal". Most of these 22 sentences follow the same structure (player led team, team was led by a player), while only one sentence raises an issue, because it has 2 player names rather than one[1].

2.3 Claim Verification

After getting all the elements we need for the verification, we build queries to look up the data tables. Since we organize the data in csv files of the same format (player_TeamA_vs_teamB.csv or team_TeamA_vs_teamB.csv), the queries share a fixed structure. Once the table name has been identified, query generation is driven by the formula obtained from the classifier. We then collect the value(s) for the check by identifying the cell values based on the key and the column predictions; such

[1] "Sergio Rodriguez was the high-point man for the 76ers, with 18 points and five assists, while Jahlil Okafor logged 20 minutes off the bench."

values are finally compared against the claim values. The claim value is translated to a numerical value if written in words (thirteen to 13), or considered True by default in case of a Boolean claim ("defeated").

Claim	Column	Formula	Row
18 points	PTS	a	Bradley Beal
five rebounds	REB	a	Bradley Beal
four assists	AST	a	Bradley Beal

Table 1: Extracted properties for an example sentence.

The following example walks through a sentence verified by our solution. Given "Bradley Beal led the way for the Wizards with a game-high 18 points, which he supplemented with five rebounds , seven assists and one steal.", we show the extracted claims and properties in Table 1.

Claim	Query Output	Evaluation
18 points	18.0	True
five rebounds	5.0	True
seven assists	4.0	False

Table 2: Evaluation of the example.

All claims require a lookup (formula is **a**) in the table player_Hornets_vs_Wizards.csv, with a primary key value **Bradley Beal** and columns **PTS**, **REB**, and **AST**, respectively. After extracting these properties, we compose and execute the final query over the table and compare with the actual claim as shown in Table 2. Our system did not identify claims such as "led" & "game-high" as our claim-identification module is limited by a regular-expression approach.

3 Results

The evaluation of our solution over the test data shows that we obtain a precision of 0.329 and a recall of 0.205. While we expected a limited recall, as we focus on a specific subset of claims among all the possible ones in the summaries, we investigated the possible causes for the low performance and found three main explanations.

Missing support for coreference resolution. In some sentences, names are not explicitly mentioned and the concerned entity is referred to as "he" or "the visiting team". As we did not implemented a specific solution for coreference resolution, this is a weak point in our system. For example, consider the sentence "It was his second double-double in a row, as he's combined for 54 points and 13 rebounds over his last two games.". The sentence is not checked by the system since no names were singled out, leading to the system missing 3 claims.

Claims requiring complex retrieval. Some statistical values are hard to fact check, such as the number of wins in the season. Consider the sentence "Over his last three games, he's combined for 34 points, 13 rebounds and five assists, while playing just 21 minutes per game.'. The verification of this claims requires to identify the last 3 games played by the player, and to sum up their scores. As another example, "Greg Monroe was the only other Laker [...]". The only way to know if he was the only other Laker to achieve something is to look at all the Lakers players scoring. These kinds of processes go beyond the ability of our data retrieval modules.

Limits in the identification of the claims. Some of the wording used in the text turned out to be hard to process both with the regex function and the Bert model. For example, all claims containing an expression like "double-double" or "6-for-13".

4 Future Work

We presented a solution for verifying statistical data claims in data summaries with limited human supervision. We believe our work shows some of the opportunities and challenges for the problem of verifying data summaries with computational methods (Saeed and Papotti, 2021). While there are some promising results for specific cases, the road to accurate and general solutions is still long. The main challenge lies in the limited amount of training data and the large variety of claim kinds, as we discuss next.

The claims in Figure 1 vary from those expressed using adjectives (e.g., strong) to others pivoting on verbs (e.g., leading). Fact-checking these claims requires to go beyond a lookup in a table, but rather, we seem to suggest the need for domain-specific rules or models, depending on the nature of the claim. For instance, for the sentence "the Grizzlies had a **strong** first half", if we interpret that the claim "strong" relates to the points scored in the first half, it means that we are interested in the sum of the first two quarters, *TEAM-PTS_QTR1* + *TEAM-PTS_QTR2*, and we have to set a threshold to the number of points starting from which the word "strong" applies. But this rule would apply

only to "strong" (and similar adjectives) used for points and the threshold would change for "fouls". In other words, we have to handle all the possible qualitative adjectives that can appear in the text for all attributes, which is challenging to do exhaustively and accurately. It is easy to see that this problem is challenging as it would either require a lot of manually defined rules or the annotations of a large number of claims to train models that handle all cases.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152, Hong Kong, China. Association for Computational Linguistics.

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification. *Proc. VLDB Endow.*, 13(11):2508–2521.

Preslav Nakov, David P. A. Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated fact-checking for assisting human fact-checkers. In *IJCAI*, pages 4826–4832. ijcai.org.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP*.

Mohammed Saeed and Paolo Papotti. 2021. Fact-checking statistical claims with tables. *IEEE Data Eng. Bull.*, 44(3).

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.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3).