Statistical Claim Checking: StatCheck in Action

Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat

Statistical Claim Checking: StatCheck in Action

Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat

Statistical Claim Checking: StatCheck in Action

Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat
firstname.lastname@inria.fr
Inria and Institut Polytechnique de Paris
Palaiseau, France

ABSTRACT

Fact-checking is a staple of journalists' work. As more and more important data is available in electronic format, computational fact-checking, leveraging digital data sources, has been gaining interest from the journalists as well as the computer science community. A particular class of interesting data sources are *statistics*, that is, numerical data compiled mostly by governments, administrations, and international organizations.

We propose to demonstrate StatCheck, a fact-checking system specialized in the French media arena. StatCheck builds upon a prior pipeline [CMT17, CMT18, DCMT19] for fact-checking statistical claims against the INSEE national statistic institute's data and reports. In collaboration with factchecking journalists from France Info (Radio France), we have revisited and improved this pipelined, and enlarged its database by an order of magnitude by adding all Eurostat statistics. StatCheck also includes two novel statistic claim extraction modules, generalizing and improving over [DCMT19]. Based on the journalists' feedback, we built a new user interface, suiting better their needs. We will showcase StatCheck on a variety of scenarios, where statistical claims made in social media are checked against our statistic data.

1 INTRODUCTION

Professional journalism work has always involved verifying information with the help of trusted sources. In recent years, the proliferation of media in which public figures make statements, including traditional media available online, as well as social media, has lead to an explosion in the amount of content that may need to be verified in order to distinguish accurate from inaccurate, and even potentially dangerous, information.

Computational fact-checking is a growing and multidisciplinary field [CLL$^+$18, NCH$^+$21] which has lead to the creation of meeting venues such as the Conference for Truth and Trust Online[1]. The main tasks of a fact-checking system are: identifying the claims made in an input document, finding the relevant evidence from a reference dataset, and optionally producing an automated verdict or if not, letting an end user decide on the truthfulness of the claim. Recent systems proposed in this area include [HZA$^+$17, KSPT20, PMYW18]. In a recent survey [SP21], the authors compare the performance of four statistical fact-checking systems [CWC$^+$20, HNM$^+$20, JTY$^+$19, KSPT20]. A statistical claim is defined as *any textual claim that can be verified over a trustworthy database by performing a mathematical expression on the database cell value* [SP21]. The four systems take in input a claim and a table, and should output the truth value of the claim.

[1] https://truthandtrustonline.com/

BDA '22, October 24–27, 2022, Clermont-Ferrand, France
2018. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

We propose to demonstrate StatCheck, a fact-checking system specialized in the French media arena. Differently from the above-mentioned systems, StatCheck also includes a claim detection step, while assuming that an end user will decide if a claim is true or not based on the evidence retrieved. Another specificity is our focus on checking statistical claims, with the help of large public statistic databases (specifically, INSEE and Eurostat); the nature and organization of these databases raises specific challenges when searching for the statistic most appropriate to check a given claim.

Architecture and outline The overall architecture of our platform is presented in Figure 1, based on which we present the outline of this paper. We first acquire statistic data from reference sources, currently INSEE and Eurostat; we describe how data is stored in Section 2. In Section 3, we show how users's keyword search queries can be answered against the stored data. To help journalists even more in their work to fact-check claims made in the public space, also ingest them in our system (by subscribing to media sources, or allowing users to upload their own content), and apply Natural Language Processing to identify, from their textual content, statistical claims. This is described in Section 4. We then describe the proposed demonstration scenarios (Section 5).

2 STATISTIC DATA ACQUISITION AND STORAGE

INSEE publishes each statistic report as an HTML page that links to **statistic tables**, which may be in Excel (the most frequent case) or in HTML. The tables are not relational. On one hand, they have human-understandable header cells not only for each column (as is the case for a relational table), but also for each line. From this perspective, a statistic file resembles more a multidimensional aggregate query result. On the other hand, many tables feature *hierarchical (nested) headers*: for instance, a header cell corresponding to "Paris (75)" may appear as a child of another header cell corresponding to "Île-de-France". To capture such structure, the data is modeled as an RDF graph, as illustrated in Figure 2; each cell becomes an RDF URI, connected to its *closest line header* (blue arrows), and *closest column header* (red arrows). The black arrows are triples relating column header cells to their parent header cell.

While revisiting the platform, we re-crawled the INSEE Web site up to May 2022, leading to 60,002 Excel files and 58,849 HTML files. We then extract statistic tables from those files and convert them into RDF graphs, accounting for 7,362,538,629 RDF triples, including 22,366,376 row or column header cells. To store this large amount of data, we use JENA TDB2[2]; loading the complete set of INSEE published statistics took around 29 hours. Subsequently, we incrementally re-crawl the Web site each night to retrieve and ingest the possible studies published every day.

[2] https://jena.apache.org/documentation/tdb2/

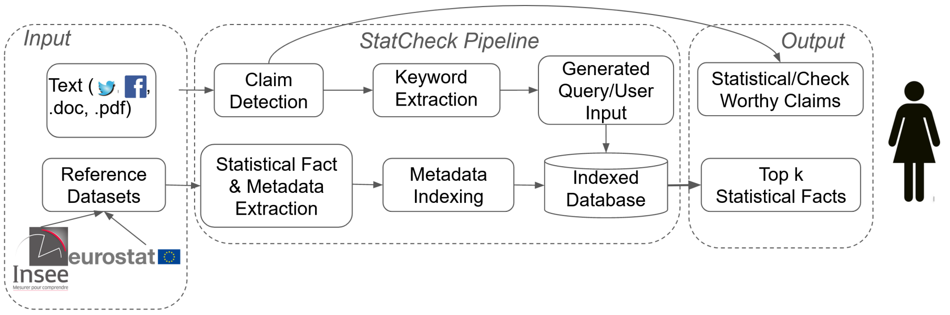
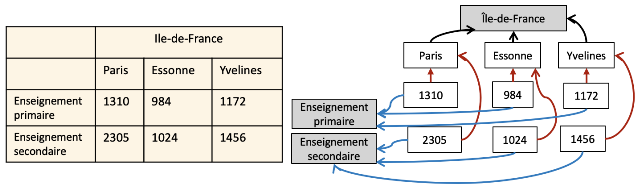


Figure 1: Statistical fact-checking architecture overview.



	Ile-de-France		
	Paris	Essonne	Yvelines
Enseignement primaire	1310	984	1172
Enseignement secondaire	2305	1024	1456

Figure 2: RDF graph modeling of INSEE table data [CMT17].

As part of our collaboration with RadioFrance, we recently added a new corpus of reference statistics, namely Eurostat: (*i*) 6,803 data tables; these are two-dimensional tables with line headers, row headers, and no nesting in the headers. Each header is a concatenation of a set of dimension values, e.g., EU15_NO as a line header corresponds to the value of metric "Union européenne - 15 pays (1995-2004) et Norvège" (*ii*) 580 dictionaries that map 243,083 statistical concepts codes, such as BLA in the above, into natural-language descriptions. The data files range from 2 lines, to 37 million lines (this largest file holds information about farmers and their agricultural properties across Europe). Together, the Eurostat data files total 414.908.786 lines. As the tables of Eurostat are normalized and flat, they do not require a semi-structured (graph) storage format. Due to their relatively high number, instead of storing them in a relational database, we keep them as plain files, and developed specialized search techniques for them, as we explain below.

3 STATISTIC SEARCH

Given a keyword query $Q = \{k_1, k_2, \ldots, k_n\}$, our purpose is to return a ranked list of the most pertinent datasets w.r.t the query. Further, if in a dataset we can identify a row, column, or a cell that seems to answer exactly the query, we return that result at the finest level of granularity.

It is important to note that *query results are virtually always numbers*, because this is what statistic institutes publish. Such numbers can only be interpreted through the *metadata* (elements of natural language) associated to them. We use $\mathcal{L} = \{T, CH, RH, C\}$ to denote the set of *possible locations* in which a term can appear in the metadata of a table, respectively: the dataset title, a column header, a row header, or a comment. The locations are important: (*i*) since a term appearing in a title is more important than one appearing in a header (Section 3.1); (*ii*) to determine whether a given dataset matches some terms in a row header and others in a column header - in which case the cell at the intersection of the row with the column likely has a very pertinent result (Section 3.2).

3.1 Dataset Indexing and Search

For each dataset d, we call **metadata of** d its *title* together with any *comment* published by INSEE next to the dataset. Further, the statistic tables within d have *row headers* and *column headers* which may be INSEE internal codes, not directly readable by non-experts. In such cases, d also includes (in a separate sheet) the *natural-language descriptions of the dimensions involved* in all these headers. These are also part the dataset's metadata.

We split the metadata of d into a set of tokens $T = \{t_1, \ldots, t_N\}$. For each token t, we identify, based on a Word2Vec model[3], the 50 tokens t' closest semantically to t. Next, for each appearance of a token t in a location l within d, our **term-location index** I_{TL} stores: (*i*) the index entry (t, d, l) corresponding to the token actually found in d; and (*ii*) 50 entries of the form $(t', d, l, dist)$, corresponding to the 50 tokens closest to t. These extra entries enable answering queries on terms close to (but disjoint from) the dataset content. For instance, when t is "*Enseignement*", t' could be "*Ecole*", "*Enseignant*", "*Elève*", etc. To determine the datasets pertinent for the query Q, we look up the query keywords in I_{TL}; a dataset containing at least one keyword is potentially interesting.

To rank datasets, a relevance score was introduced in [CMT18], based on word distances between the query keywords and the datasets' metadata, and also reflecting the locations where the keywords were found for each dataset. To this custom score, we have added the classic BM25 [RZ09] computed over all the datasetes' metadata, and are currently experimenting with the two.

3.1.1 *Improving the INSEE pipeline.*

We brought several improvements over [CMT18]. First, we collected and processed metadata during data acquisition (as opposed to retrieving it from Fuseki after loading). Tokening and lemmatization, based on Spacy, were sped up by accumulating all the metadata of a table in a single string for which we call Spacy only once per dataset.

Next, we improved the understanding and interpretation of geographical query terms. Users may query for a city, department, such as "Essonne", or region, while statistic data may be available at different levels of aggregation, e.g., a dataset may be about "Île-de-France" (the region). The system needs to be aware of the relationship between the two, in order to also consider returning data about the region. To that purpose, we used a list[4] containing

[3]We used the model frWac_non_lem_no_postag_no_phrase_200_skip_cut100.bin from https://fauconnier.github.io

[4]https://www.data.gouv.fr/fr/datasets/regions-departements-villes-et-villages-de-france-et-doutre-mer/

the names of all cities, departments and regions of France (35.984 names in all). To quickly identify them in user queries, we used an implementation[5] of the FlashText algorithm [Sin17], capable of finding, in a document of size N, one of M fixed keywords, in $O(N)$ time complexity. This is much faster than the typical $O(NM)$ cost of regular expression pattern matching, a sizable advantage for us. The optimizations introduced above allow us to reduce the overall INSEE loading and indexing from 39 hours to about 3h.

3.1.2 Eurostat indexing pipeline. As detailed in Section 2, Eurostat datasets (*i*) have a simpler tabular structure, (*ii*) together, are much larger than the INSEE corpus, leading to metadata 100× larger, and (*iii*) contain a significant number of very large files (hundreds of millions of rows). This discourages row- or cell-level indexing of the Eurostat metadata, as the index would be much too large.

Instead, we decide to use Eurostat statistical concept codes as a basis for indexing, as follows. Let c be a Eurostat concept, e.g., ED1, appearing in dataset d at a location $l \in \mathcal{L}$. We insert in a **concept-dataset index** I_C the entry (c, d, l). For instance, if c appears in 1 million row headers in d, only one entry with $l = RH$ is generated.

Next, let d_c be the natural-language description that Eurostat associates to c, e.g., "*Enseignement primaire*" for ED1. Let $T = \{t_1, t_2, \ldots, t_N\}$ be the tokens in d_c, and for $1 \leq i \leq N$, let t_i^j, for $1 \leq j \leq 50$, be the closest tokens to t_i. For each $t_i \in T$, we look up all the (c, d, l) pairs in I_C, and insert in the **term-dataset index** I_T, a (t_i, d, l) entry. Next, for every t_i^j similar to t_i, we insert in I_T an entry $(t_i^j, d, l, dist, t_i)$, where $dist$ is the distance between t_i and t_i^j. I_C and I_T are stored in Redis; I_C only serves to accelerate the construction of I_T, which was prohibitively slow without it. Indexing the complete Eurostat data in this way took around 4 minutes.

When a query $\{k_1, \ldots, k_m\}$ is asked, we search I_T for entries of the form (k_i, d, l) or $(k_i, d, l, dist, k_i')$. Any dataset having an entry for at least one k_i is potentially interesting; then, we score them either using either BM25 or the score introduced in [CMT18].

3.2 Data Cell Indexing and Search

From the 20 highest-scores datasets, we try to extract results *at the cell, row, or column* level. Given the query $\{k_1, \ldots, k_m\}$, the index I_{CL} (INSEE) or I_T (Eurostat) returns, for each $1 \leq i \leq m$, a set of entries, each containing: a keyword k_i or a close term k_i^j, a dataset d, and a location l. Let d be one of the most interesting datasets, and $I(d)$ be the set of all index entries for this query and d.

If $I(d)$ only features title (T) or comment (C) locations, then d is pertinent as a whole, and no cell search is needed.

On the contrary, if $I(d)$ has some entries with $l = RH$, say, for keyword k_1 (or a close term k_1^j), and other entries with $l = CH$, say, for k_2 (or some k_2^j), then at the intersection of the row whose header matches k_1, with the column whose header marched k_2, is a cell that appears to be a very fine-granularity answer to Q.

- If d is an INSEE dataset, $I(d)$ specifies exactly which rows and columns are concerned. Then, the cell is identified by asking a SPARQL query [CMT18], evaluated by Fuseki.
- On the contrary, if d is an Eurostat dataset, $I(d)$ only specifies that "*some* row (column) headers match". Identifying the relevant cells require more effort, as we explain below.

[5]https://github.com/vi3k6i5/flashtext

Identifying the right columns An Eurostat file has at most a few dozen columns. To find the column referred to by an $I(d)$ entry whose key is k, we read the first (header) line of d, and identify the column(s) whose header contains k. This step is very fast, because we only read the beginning of the file.
Identifying the right rows This required an extra index to be efficient even on files with millions of rows. Specifically, when indexing the data, we also create a **row location index** I_R which, for every *row header* entry $(k, d, RH) \in I_T$, stores $(k, d, R_{k,d})$, where $R_{k,d}$ is the list of the indexes of data rows in d which contain k in their header. For efficiency, I_R is saved directly as a file on disk, and supports direct access by k and d, following the Adaptive Positional Map of [ABB+15].
Cell extraction Once the right rows and column indexes are known, we read the relevant rows from d, and carve out of them the relevant data cell(s). With the help of a line cache library[6], this process is quite efficient.

Overall, on INSEE datasets, using Fuseki, data cell search takes 35ms up to 2.86s. On Eurostat, using I_R, we record 4.76μs up to 2.66s. The lower bound is higher for INSEE because we have to pass SPARQL queries across a connection to the Fuseki server.

4 CLAIM DETECTION

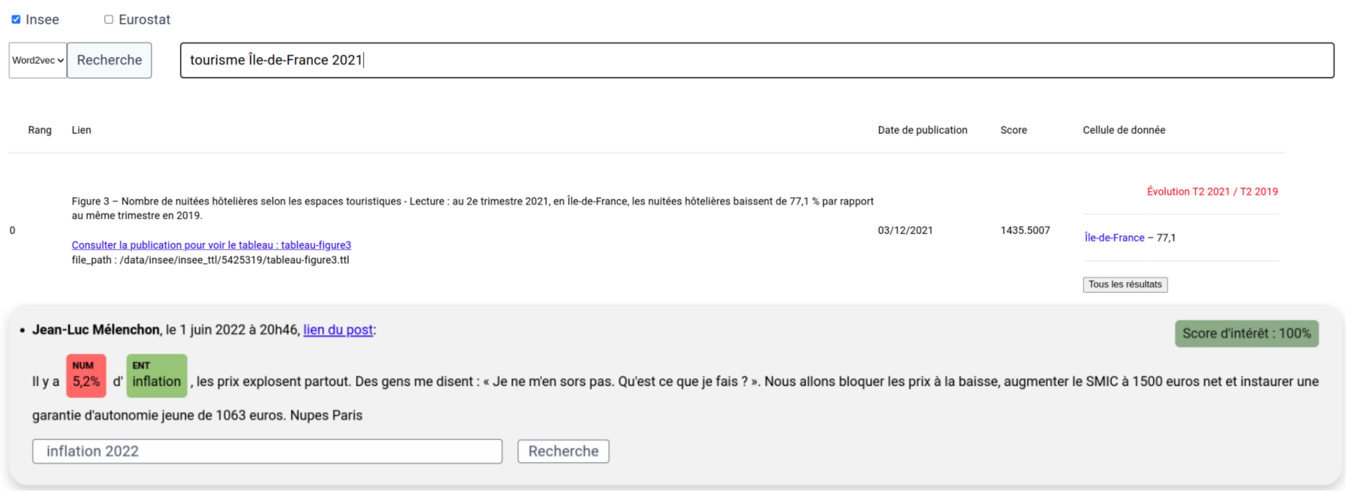
A claim is a statement to be validated, that is we aim to establish if it is true of false. The validation is achieved by finding related statements, called evidence, which back up or disprove the claim. In our work, the claims are detected in an input text, while the evidence is retrieved from a set of trusted sources, our reference datasets. Our platform detects claims from text stored in *.txt*, *.odt*, *.docx* or *.pdf* files, and from the Twitter and Facebook posts of public figures. For posts, our platform retrieves regularly the most recent updates of a predefined group of users.

4.1 Statistical Claim Detection

In [DCMT19], the authors introduced a statistical claim detection method that given an input set of statistical entities (e.g. chômage, coefficient budgétaire) and a sentence, it retrieves all the *statistical statements* of the form ⟨`statistical entity, numerical value and unit, date`⟩ present in the sentence. *The statistical statement, if present, represents the statistical claim to be verified.* The statistical entities and units are retrieved using exact string matching, while the date is extracted using HeidelTime [SG10], a time expression parser. If no date is found by the parser, the posting timestamp is used. More context about the claim to be verified is found using a Named Entity Recognition (NER) model, which returns organizations and locations. We note, however, that the organization and location are optional, while a statistical statement is not complete without one of its three elements. The initial statistical entity list is constructed from the reference datasets by taking groups of tokens from the headers of tabels, we refer to [DCMT19] for more details.

We improved the method presented in [DCMT19] to optimize both speed and quality of extractions. We refer to the two methods as OriginalStatClaim [DCMT19] and StatClaim. We first performed a more careful match between the tokens of a sentence and our input statistical entities. Using the syntactic tree of the sentence and a lemmatizer, statistical entities are matched on using their

[6]https://docs.python.org/fr/3/library/linecache.html



☑ Insee ☐ Eurostat

Word2vec | Recherche | tourisme Île-de-France 2021

Rang	Lien	Date de publication	Score	Cellule de donnée
0	Figure 3 – Nombre de nuitées hôtelières selon les espaces touristiques - Lecture : au 2e trimestre 2021, en Île-de-France, les nuitées hôtelières baissent de 77,1 % par rapport au même trimestre en 2019. Consulter la publication pour voir le tableau : tableau-figure3 file_path : /data/insee/insee_ttl/5425319/tableau-figure3.ttl	03/12/2021	1435.5007	Évolution T2 2021 / T2 2019 Île-de-France – 77,1 Tous les résultats

- **Jean-Luc Mélenchon**, le 1 juin 2022 à 20h46, lien du post:

Score d'intérêt : 100%

Il y a NUM 5,2% d' ENT inflation , les prix explosent partout. Des gens me disent : « Je ne m'en sors pas. Qu'est ce que je fais ? ». Nous allons bloquer les prix à la baisse, augmenter le SMIC à 1500 euros net et instaurer une garantie d'autonomie jeune de 1063 euros. Nupes Paris

inflation 2022 | Recherche

Figure 3: Screen captures of our tool. Top: sample candidate data cells with the corresponding header row (blue) and header column (red); bottom: tweet analysis interface.

lemma, and are extended to contain the entire nominal group of the matched token. Numerical values are associated with units using both lemma matching from our set of units and a syntactic analysis. Units can be a noun following a numerical value, or a nominal group containing one or more units. (e.g. "millions d'euros"). As in the original approach, if we retrieve a statistical statement of the form ⟨`statistical entity, numerical value and unit, date`⟩, we have found a claim to verify. In the default setting of our algorithm, a claim should contain all the three element. In addition, we filter claims coming from sentences in which the verb is in the future tense or in the first person. We have incorporated feedback from the journalists and we allow through our interface a relaxation of the approach: the numerical value can be missing, and the filtering on the verbs can be removed.

4.2 Check-worthy Claim Detection

To complement the statistical claim detection model, we developed a model that is not conditioned on a set of initial statistical entities. The model classifies a sentence as check-worthy or not, where check-worthiness is defined as *sentences containing factual claims that the general public will be interested in learning about their veracity* [AHLT20]. We leveraged the ClaimBuster dataset [AHLT20], a English dataset containing check worthy claims from the U.S. Presidential debates, to train a cross-lingual language model, XLM-R [CKG+19], which is able to perform zero-shot classification on French sentences after having been trained on English data.

The ClaimBuster dataset ClaimBuster is a crowd-sourced dataset where the sentences from the 15 U.S. presidential elections debates from 1960 to 2016 have been annotated. The labels are Non-Factual Sentences (NFS), Unimportant Factual Sentences (UFS) or Check-Worthy Factual Sentences (CFS). The dataset contains $23K$ sentences, and a subset of higher quality of $11K$ sentences was produced by the authors for training models on classification tasks. In this smaller dataset, the NFS and UFS labels are grouped together as negative labels, and the CFS labels are considered positive. We chose this higher quality dataset to fine-tune the XLM-R model.

Dataset	Precision	Recall	F1 score	Accuracy
ClaimBuster	0.883	0.848	0.865	0.934
French tweets	0.612	0.769	0.682	0.856

Table 1: Evaluation of the fine-tuned XLM-R model.

Fine-tuning the XLM-R model The XLM-R model is a Transformer-based masked language model trained on one hundred languages with 2.5TB of Common Crawl data. It achieves state-of-the-art results on multilingual tasks such as the XNLI benchmark [CRL+18], while remaining competitive on monolingual tasks. We used a pre-trained model with a vocabulary size of $250K$, 12 hidden layers of size 768 and 12 attention heads. To account for the unbalanced ratio of labels, we used a weighted cross-entropy loss. The dataset was split into train, dev and test datasets with a ratio of 80%/%10%/10%.

Evaluation To ptimize the performance, we trained the model with different hyperparameters. The best results were obtained with a learning rate of $5 \cdot 10^{-5}$, a batch size of 64, and using the AdamW optimizer. To evaluate the performance of the different models on French data, we annotated 200 randomly sampled French tweets and labeled them as check-worthy or not following the definition in [AHLT20]. Two annotators labeled each tweet; in the golden standard, a tweet is deemed check-worthy if both annotators agree on it, and not check-worthy otherwise. The Cohen Kappa score for inter-annotator agreement is 0.6, which is considered moderate to substantial agreement. The results can be found in Table 1. The drop in precision on French data could be because we are evaluating on a small test dataset (only 200 tweets with 39 positive examples), or the fact that the tweets' format and vocabulary might be different than the ones in the Presidential debate sentences used for training.

4.3 Integration and Evaluation of the Claim Detection Models

We evaluate the claim detection models, *(OriginalStatClaim [DCMT19], StatClaim and CheckWorthyClaim)*, on a set of 1595 tweets. Each tweet was labeled with two classes: *"Verifiable numerical claim"* (True if the tweet contains at least one numerical and verifiable claim") and *"INSEE statistical claim"* (True if the tweet contains at

Models	Precision	Recall	F1 score
OriginalStatClaim	0.692	0.466	0.557
StatClaim	**0.833**	0.517	0.638
CheckWorthyClaim	0.701	**0.915**	**0.794**

Table 2: Model evaluation on verifiable numerical claims.

Models	Precision	Recall	F1 score
OriginalStatClaim	0.282	0.688	0.400
StatClaim	**0.333**	0.750	**0.462**
CheckWorthyClaim	0.195	**0.938**	0.323

Table 3: Model evaluation on INSEE statistical claims.

least one numerical, statistical claim verifiable against the INSEE dataset"). We chose these two labels as the first one gives us an indication of the tweets that can be verified if we had an unlimited access to resources, while the second class identifies the tweets verifiable in the setting in which we have access to only one resource. To construct our set, we gathered 1595 random tweets from our scraped dataset. Then, we automatically detected if a tweet contained a numerical value, if not, the tweet was labeled as negative for both classes. After that first step, we manually labeled the remaining 101 tweets. Two annotators labeled each tweet, and the gold standard was chose as True if both annotators agreed. For the class *"verifiable numerical claim"*, we obtained a Kappa inter-Annotator Agreement score of 0.917 (almost perfect agreement) and *59* tweets were labeled as positive. For the class *"INSEE statistical claim"* we obtained an inter-annotator Agreement score of 0.807 (subtantial agreement) and *16* tweets were labeled as positive.

Evaluation procedure For StatClaim and OriginalStatClaim, a tweet is considered positive if models return at least one extracted statistical statement. Our StatClaim was used in its default configuration: extractions with numerical values, and without verbs conjugated in the future or at the first person. For CheckWorthyClaim, a tweet is considered positive if the model return a check-worthy score > 0.9. We report the results in Table 2 and Table 3. StatClaim performs better than the original at detecting INSEE verifiable claims, and CheckWorthyClaim vastly outperforms both models on the detection of numerical claims, as they are a subset of check-worthy sentences that the model was trained to detect.

Default claim detection strategy. By default, StatCheck uses StatClaim for statistical claim detection. However, given the good performance of CheckWorthyClaim on numerical claims, we allow users to switch to it, even if we might not be able to verify them against the reference datasets.

5 DEMONSTRATION SCENARIOS

Our system is developed in Python and deployed on a Unix server. Its GUI is accessible via a Web server; Figure 3 shows two screen captures. Demonstration attendees will be able to:

- Ask queries in the statistic search interface, and inspect the results, at the level of cell, line, or column, together with their metadata from the original statistic site (INSEE or Eurostat);
- Visualize incoming social media messages (as they arrive in real-time and are stored and analyzed by our platform), in order to see (*i*) the statistical mentions and (*ii*) claims deemed potentially check-worthy, identified in these messages, as illustrated in Figure 3. Note that the system also proposes candidate search queries for the statistic search interface.
- Users will be able to select various options (restrict to numerical claims or not, include statements about the future or not, include first-person texts or not, etc.) and see their impact on the claim extraction output.
- Write their own text and/or suggest other content to be processed by our analysis pipeline (Section 4).

ACKNOWLEDGEMENTS

We thank Radio France, France Info, and in particular Antoine Krempf and his *Le vrai du faux* team, for their valuable feedback. This work is partially funded by AI Chair SourcesSay project (ANR-20-CHIA-0015-01).

REFERENCES

[ABB+15] Ioannis Alagiannis, Renata Borovica-Gajic, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. Nodb: efficient query execution on raw data files. *Commun. ACM*, 58(12):112–121, 2015.

[AHLT20] Fatma Arslan, Naeemul Hassan, Chengkai Li, and Mark Tremayne. A Benchmark Dataset of Check-worthy Factual Claims. In *14th International AAAI Conference on Web and Social Media*. AAAI, 2020.

[CKG+19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[CLL+18] Sylvie Cazalens, Philippe Lamarre, Julien Leblay, Ioana Manolescu, and Xavier Tannier. A content management perspective on fact-checking. In *WWW (Companion Volume)*, pages 565–574. ACM, 2018.

[CMT17] Tien Duc Cao, Ioana Manolescu, and Xavier Tannier. Extracting linked data from statistic spreadsheets. In *International Workshop on Semantic Big Data*, International Workshop on Semantic Big Data, pages 1 – 5, 2017.

[CMT18] Tien-Duc Cao, Ioana Manolescu, and Xavier Tannier. Searching for Truth in a Database of Statistics. In *WebDB*, pages 1–6, 2018.

[CRL+18] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *EMNLP*, 2018.

[CWC+20] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. Tabfact : A large-scale dataset for table-based fact verification. In *ICLR*, 2020.

[DCMT19] Tien Duc Cao, Ioana Manolescu, and Xavier Tannier. Extracting statistical mentions from textual claims to provide trusted content. In *NLDB*, 2019.

[HNM+20] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *ACL*, pages 4320–4333, 2020.

[HZA+17] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, Vikas Sable, Chengkai Li, and Mark Tremayne. Claimbuster: The first-ever end-to-end fact-checking system. *PVLDB*, 10(12):1945–1948, 2017.

[JTY+19] Saehan Jo, Immanuel Trummer, Weicheng Yu, Xuezhi Wang, Cong Yu, Daniel Liu, and Niyati Mehta. Verifying text summaries of relational data sets. In *SIGMOD*, SIGMOD '19, page 299–316, 2019.

[KSPT20] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. Scrutinizer: Fact checking statistical claims. *Proc. VLDB Endow.*, 13(12):2965–2968, 2020.

[NCH+21] Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. Automated fact-checking for assisting human fact-checkers. In *IJCAI*, pages 4551–4558, 2021. Survey Track.

[PMYW18] Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. DeClarE: Debunking fake news and false claims using evidence-aware deep learning. In *EMNLP*, pages 22–32, 2018.

[RZ09] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond.* Now Publishers Inc, 2009.

[SG10] Jannik Strötgen and Michael Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Int'l. Workshop on Semantic Evaluation*, pages 321–324, 2010.

[Sin17] Vikash Singh. Replace or retrieve keywords in documents at scale. *CoRR*, abs/1711.00046, 2017.

[SP21] Mohammed Saeed and Paolo Papotti. Fact-checking statistical claims with tables. IEEE Data Engineering Bulletin, 2021.