# Mining an “Anti-Knowledge Base” from Wikipedia Updates with Applications to Fact Checking and Beyond

Georgios Karagiannis, Immanuel Trummer, Saehan Jo, Shubham Khandelwal
Cornell University, NY, USA
{gk446,it224,sj683,sk3266}@cornell.edu

Xuezhi Wang, Cong Yu
Google, NY, USA
{xuezhiw,congyu}@google.com

## ABSTRACT

We introduce the problem of anti-knowledge mining. Our goal is to create an “anti-knowledge base” that contains factual mistakes. The resulting data can be used for analysis, training, and benchmarking in the research domain of automated fact checking. Prior data sets feature manually generated fact checks of famous misclaims. Instead, we focus on the long tail of factual mistakes made by Web authors, ranging from erroneous sports results to incorrect capitals.

We mine mistakes automatically, by an unsupervised approach, from Wikipedia updates that correct factual mistakes. Identifying such updates (only a small fraction of the total number of updates) is one of the primary challenges. We mine anti-knowledge by a multi-step pipeline. First, we filter out candidate updates via several simple heuristics. Next, we correlate Wikipedia updates with other statements made on the Web. Using claim occurrence frequencies as input to a probabilistic model, we infer the likelihood of corrections via an iterative expectation-maximization approach. Finally, we extract mistakes in the form of subject-predicate-object triples and rank them according to several criteria. Our end result is a data set containing over 110,000 ranked mistakes with a precision of 85% in the top 1% and a precision of over 60% in the top 25%. We demonstrate that baselines achieve significantly lower precision. Also, we exploit our data to verify several hypothesis on why users make mistakes. We finally show that the AKB can be used to find mistakes on the entire Web.

**PVLDB Reference Format:**
Georgios Karagiannis, Immanuel Trummer, Saehan Jo, Shubham Khandelwal, Xuezhi Wang, Cong Yu. VLDB. *PVLDB*, 13(4): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/3372716.3372727

## 1. INTRODUCTION

Constructing a knowledge base of high quality facts at large scale is one of the main topics of many research communities ranging from natural language processing to semantic web and data mining. The research work has been impactful—leading to significant advancement in major search engine features such as Google’s Knowledge Graph [26] and Bing’s Satori [19]. Little attention, however, has been paid to the large number of erroneous facts on the Web. While storing all the world’s wrong facts would be infeasible and not particularly useful, we argue that a knowledge base of *common* factual mistakes, what we call an *anti-knowledge base*, can complement the traditional knowledge base for many applications including, but not limited to, automated fact checking. Specifically, collecting common factual mistakes enables us to develop learning algorithms for detecting long tail factual mistakes in a much more precise way than verification against a traditional knowledge base can accomplish. Furthermore, an anti-knowledge base can provide values for the research community in conducting new studies such as how users make mistakes and designing novel systems such as alerting users when they are working on mistake-prone topics.

In this paper, we formally introduce the concept of an anti-knowledge base (AKB) and propose the first algorithmic approach for *anti-knowledge mining* (AKM). Similar to traditional knowledge bases, an anti-knowledge base contains ⟨subject, predicate, object⟩ triples with associated metadata, where each triple represents a factually mistaken claim that has been asserted by users on the Web. Whenever possible, we pair each *mistaken* triple with a correction triple in the anti-knowledge base so that users get the correct knowledge as well. A formal and more elaborate definitions can be found in Section 3.

At first, it may seem possible to mine mistakes by comparing claims extracted from text against an existing knowledge base. As we will show in Section 8, this approach leads however to an excessive number of false positives (i.e., claims erroneously marked up as mistakes) due to incomplete knowledge bases and imperfect extraction. We could leverage prior work on detecting conflicting statements on the Web [28]. However, conflicting statements may simply indicate a controversial topic where different users have different opinions, rather than a factual mistake. Our key idea is to exploit a property that distinguishes factual mistakes from generally accepted claims as well as subjective topics: a diligent user community will *correct* factual mistakes once they are spotted.

A static view on the Web is insufficient to leverage this property. Hence, we assume that we are given an update log on a collection of documents. By comparing original and updated text versions for each update, we obtain can-

*Proceedings of the VLDB Endowment,* Vol. 13, No. 4
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3372716.3372727

didates for factual mistakes. Of course, text updates may not only correct factual mistakes (but also expand text, correct spelling mistakes, or update controversial statements to reflect one author's personal opinion). One of the main challenges solved by our approach is therefore to filter out those updates that correct factual mistakes to form AKB entries. To do so, we exploit a multi-stage pipeline combining simple heuristics, a probabilistic model trained via unsupervised learning, as well as feedback obtained from a static Web snapshot (counting the number of supporting statements for text before and after updates).

**Example 1.1.** Imagine the sentence "Heineken is Danish" is updated to "Heineken is Dutch". This update is correcting a factual mistake since Heineken is indeed based in the Netherlands (as can be learned from authoritative sources such as the company's Web site). Our goal is to distinguish such updates from other types of updates. We use this example (an actual update mined by our approach) throughout most of the paper.

For this paper, we leverage the Wikipedia update log as source for factual mistake candidates (while we exploit a snapshot of the entire Web for feedback). We select Wikipedia due to its size, breadth of topics, and due to its large and active user community. Furthermore, mistakes that appear on Wikipedia tend to propagate further (we show some examples for mistakes propagating from Wikipedia to the rest of the Web in Section 8). Therefore, the potential impact (e.g., via fact checking tools) of mining mistakes from Wikipedia is higher than for other Web sources. While we focus on Wikipedia in this paper, please note that the general approach we propose is not specific to Wikipedia and could also be applied to different sources of updates and static feedback (e.g., an update log of Web documents in a company's intranet). Finally, note that Wikipedia is imperfect and may contain uncorrected mistakes. We will not be able to spot those mistakes by mining the update log. However, our goal is to mine common factual mistakes. Each occurrence increases the likelihood that at least one corresponding correction appears in the update log. The fact that our approach works best for mistakes that are corrected at least once is therefore in line with our overall goals.

We summarize our original scientific contributions:

- We introduce the problem of mining factual mistakes from dynamic document collections.
- We present a first corresponding approach that uses a multi-step pipeline and is able to handle large amounts of input data.
- We provide extensive experimental results, analyzing the quality and other properties of the provided data set and demonstrating that our mining algorithms are efficient and effective.

The remainder of this paper is organized as follows. We discuss differences to prior work in Section 2. In Section 3, we formalize our problem model and terminology. We give a high-level overview of our mining pipeline in Section 4. In Sections 5, 6 and 7, we describe each of the three pipeline stages in detail. In Section 8, we analyze our approach and the generated data set. We also compare against baselines.

## 2. RELATED WORK

Prior efforts [29] at generating corpora for fact checking based on the ClaimReview standard [24] leverage manual annotations by human fact checkers. This limits coverage and size of generated data sets. Prior work on mining conflicting Web sources has focused on extracting correct facts [31] or subjective opinions [28] as opposed to mining factual mistakes with high precision. A large body of work in the area of fact checking is aimed at supporting human fact checkers in verifying specific claims or documents [9–13], e.g. by identifying relevant text documents for verification [29]. The proposed approach is fully automated and targeted at mining factual mistakes from large document collections. Note that tools such as ClaimBuster [11,12] link claims to verify to previously annotated fact checks. Automatically generated fact checks may extend the coverage of tools working according to that principle.

Knowledge Base Construction has received wide attention in both academia and industry [7,18,25,27,33]. Open information extraction is a closely related area, where the goal is to extract relational tuples from massive corpora without a pre-specified vocabulary [1,8]). Our work is complementary as we mine *anti*-knowledge. We apply our approach to Wikipedia update logs. In that, our work connects to prior work leveraging Wikipedia as data source (e.g., identifying relevant articles to controversial topics [23], identifying update intentions [30], or mining data sets for grammatical error corrections [16]). The goal of our work (as well as the method) differs however from all of those. Our work is also complementary to prior work leveraging different types of data from the Wikipedia corpus (e.g., analyzing logs of structured Wikidata queries [3], as opposed to logs of updates to natural language Wikipedia text).

Our approach uses a probabilistic model and unsupervised learning (in one stage of the pipeline) to classify updated sentences as factual mistakes, based on several input features. In that, it connects in general to systems performing weakly supervised learning such as Snorkel [20] and approaches for efficient inference, e.g. based on Gibbs sampling [5]. Compared to those generic approaches and systems, our contribution lies in developing a probabilistic model (among other pipeline stages) that is specialized to the problem we are addressing. We also compare against Snorkel in multiple variants in Section 8.

We leverage input from the crowd (i.e., Wikipedia authors and Web authors) to identify factual mistakes. This connects to prior work on inferring truth from crowd input [2,4,15,17,21,32]. We compare against multiple algorithms from this line of work in Section 8. Finally, our probabilistic model uses two separate data sources, update logs and Web hits. In that, it connects to other work leveraging different input sources to come to more reliable conclusions, e.g. auto-completion and click logs [14]. Those prior approaches are not directly applicable to our scenario as input (update logs and Web claims instead of query and click logs) and desired output are different.

## 3. DEFINITIONS

We introduce terminology used throughout the paper.

**Definition 3.1.** An **Update** is in this paper a pair of sentences, $\langle s_1, s_2 \rangle$, such that $s_1$ was changed to $s_2$ by a Web author. A **Correcting Update** is in update where $s_1$ contains a factual mistake while $s_2$ does not.

**Definition 3.2.** An **Entity-Swap Update** is an update $\langle s_1, s_2 \rangle$ where the only difference between $s_1$ and $s_2$ is that one entity is replaced by another one. We consider entities that can be resolved against entries in a large knowledge base. A **Number-Swap Update** is an update where only one number is replaced by another number.

**Example 3.1.** The update used as running example is an entity-swap update as only the country entity is replaced.

**Definition 3.3.** An **Anti-Knowledge (AK) Triple** is a subject-predicate-object triple $\langle S, P, O \rangle$. Its semantics is a claim that associates a property (the object) with the subject over the predicate. This claim is however factually incorrect, i.e. it contradicts a generally accepted ground truth.

**Definition 3.4.** A **Correction** to an AK triple $\langle S_a, P_a, O_a \rangle$ is another subject-predicate-object triple $\langle S_c, P_c, O_c$ with the following properties. First, the correction changes exactly one of the three components (subject, predicate, or object) compared to the AK triple (i.e., either $S_a \neq S_c$ or $P_a \neq P_c$ or $O_a \neq O_c$). Second, the claim represented by the correction is factually correct.

**Definition 3.5.** Each pair of an AK triple and a correction is associated with **Meta-Data** of the form $\langle u, h \rangle$ where $u = \langle s_1, s_2 \rangle$ is the update a triple was extracted from, $h = \langle h_O, h_U \rangle$ is the number of times the original sentence $s_1$ and the updated sentence $s_2$ appears on the Web.

**Definition 3.6.** An **Anti-Knowledge Base (AKB) Entry** is a tuple of the form $\langle t, c, m \rangle$ where $t$ is an AK triple, $c$ its correction, and $m$ associated meta-data. An **Anti-Knowledge Base** is a collection of AKB entries.

This leads to the following, novel, variant of knowledge mining.

**Definition 3.7.** The **Anti-Knowledge Mining** problem is defined as follows. We are given as input a set $U$ of updates (or, alternatively, a time sequence of Web snapshots from which updates are extracted). Also, we are given a document collection $D$ that can be used to correlate updates in $U$. Given $\langle U, D \rangle$, the goal is to mine an AKB.

## 4. OVERVIEW OF MINING APPROACH

Figure 1 shows an overview of the AKB extraction pipeline. The input to the pipeline is a set of sentence pairs, representing updates to a document collection (e.g., Wikipedia). The output is a list of triples representing factual mistakes (together with associated meta-data, e.g. the sentences they have been extracted from). Those triples are extracted from sentence updates we believe to represent factual corrections. We rank output triples by their likelihood to describe mistakes without ambiguity.

There are many reasons to update document collections such as Wikipedia. Updates may simply polish the formulation, make a given statement more precise or extend it. We describe the filtering stage in more detail in Section 5.

Next, we use information extraction methods to extract SPO triples from those updates. We extract only from those triples that have been kept during the initial filtering step. Our ultimate goal is to obtain triples representing mistakes.

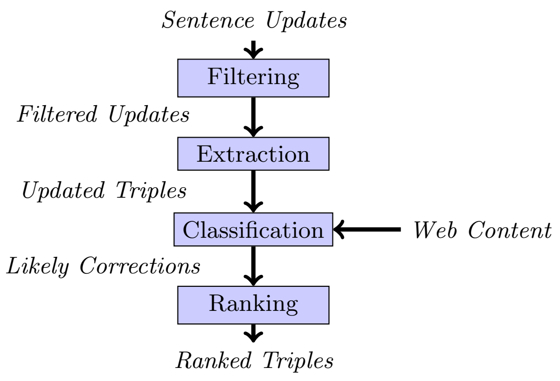

**Figure 1: Overview of the AKB mining pipeline.**

Hence, we focus extraction on those sentence parts that have been changed by correcting updates.

The triples extracted at this point still contain a high percentage of correct triples (which we are not interested in). We use unsupervised learning to classify triples as AKB candidates. Our probabilistic model is seeded by correlating updates with Web content. Statements that appear more frequently on the Web are less likely to be erroneous. The final classification is based on several factors, including occurrence frequencies but also natural language models. Section 6 describes the classification approach in detail.

The resulting triples vary in quality. Typically, a large fraction of result triples lack relevant context or are ambiguous, among other problems. We therefore use several ranking criteria in order to associate triples with quality scores. Ranking gives AKB users the possibility to trade precision for recall by selecting quality thresholds. To calculate quality scores, we consider the extraction set holistically and exploit similarities between triples for ranking. The ranking methodology is described in detail in Section 7.

## 5. FILTERING UPDATES

The input to our pipeline is the Wikipedia update stream. More precisely, we obtain for each Wikipedia page a sequence of page versions. We consider the last 1,000 updates for each page on the English version of Wikipedia. Our goal is to identify those updates that correct factual mistakes and to mine mistake and corrected version.

We focus on claims that link an entity to a property (e.g., another entity or a numerical property). Such claims are common and are the typical focus of prior work in knowledge mining. They can be represented as subject-predicate-object triples. Hence, the output of the first pipeline stage are pairs of triples, extracted from updates. The first triple is extracted from the Wikipedia page version before the update, the second triple is the corresponding triple after the change. We are interested in triple pairs where the second triple is a correction of the first.

Next, we describe the sequence of heuristics that is used to identify promising updates. We initially compare for each update prior and updated page version. We compare both version sentence by sentence and retain sentences that match the two patterns (entity swaps and number swaps) described in Section 3. Considering sentence pairs with a small delta makes a correcting update more likely. It reduces the chance that the purpose of the update was a reformulation or replacing prior (presumably correct) information with new information. Also, if the update is a correction, both considered patterns make it easy to identify the correction focus.

We are not interested in updates that correct spelling mistakes (as our goal is to support research on fact checking). Hence, for entity swaps, we compare prior and updated version via the Levenshtein distance. If the distance is below a threshold, we discard the corresponding update. Updates that replace words by synonyms are no corrections. We use WordNet to filter out synonym updates. Another common type of update replaces one entity by a specialization (to make the previous sentence version more precise). This type of update often applies to entities that represent geographic regions. We use geographic containment relationships, taken from the Google knowledge graph, to discard corresponding updates. On the remaining updates, we use the BERT [6] language model, fine-tuned on a Natural Language Inference task, to identify updates where prior and new version contradict each other (a prerequisite for a factual mistake correction).

We observed a large number of Wikipedia updates on controversial topics (e.g., concerning the attribution of certain geographic areas in case of territorial conflicts). In such cases, groups of Wikipedia authors mutually revert each other's changes. We discard such cases as it is difficult to identify the ground truth from updates (there may not even be a generally accepted ground truth). We therefore compare all remaining updates and only keep the ones that are uncontested (i.e., the corresponding update was never reverted in the update stream). We use information extraction to extract triples from before and after the update. Triple pairs that were affected by the update are passed on to the next pipeline stages.

**Example 5.1.** The running example update passes the heuristic filters (e.g., the edit distance between Danish and Dutch is sufficiently high and Dutch is not a specialization of Danish). Hence, the output of the filtering stage will contain corresponding triples ($\langle Heineken, is, Danish\rangle$ and $\langle Heineken, is, Dutch\rangle$).

## 6. CLASSIFYING UPDATES

We apply a probabilistic model to assess the probability that a given update represents the correction of a factual mistake.

### 6.1 Model Overview

The goal of the model is to determine the likelihood that a given update corrects a factual mistake. We can mine anti-knowledge from the original triple in such updates.

The scale and diversity of our input data motivates a completely unsupervised approach. The structure of our model is based on commonsense principles (e.g., correct statements are more likely than incorrect statements). The model contains parameters whose values we cannot know a priori (e.g., the probability that an average Wikipedia author makes a mistake). We therefore apply an expectation-maximization approach, that learns model parameters and likely values for random variables in an iterative approach.

Using Wikipedia updates as model input is insufficient. For instance, how can we determine whether a given update corrects or introduces a factual mistake? Hence, we use

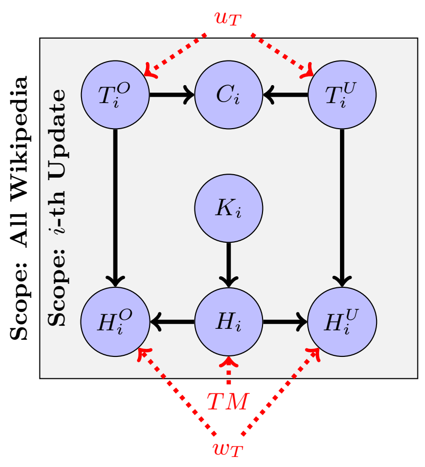

**Figure 2: Overview of probabilistic model.**

the entire Web text as a (noisy) source of feedback. Given sentence pairs (before and after an update), we mine the Web to count the number of occurrences for each of them. The number of occurrences forms one out of several input features for our model.

We deliberately use the number of sentence occurrences instead of the number of triple occurrences as input (even though we are ultimately interested in finding triples representing factual mistakes). Considering entire sentences instead of shorter triples reduces the chances of accidental matches. For instance, counting occurrences for the triple ⟨Michael Jordan,born in,1963⟩ is likely to include erroneous matches (for the American baseball player of that name), if the triple was extracted from the sentence "Michael Jordan, the English goalkeeper, was born in 1983".

### 6.2 Detailed Description

Figure 2 shows an overview of our probabilistic model. We introduce several random variables for each updated triple. For the $i$-th update, we binary random variable $T_i^O$, indicating whether the original statement was true, and $T_i^U$ indicating whether the updated statement is factually correct. We associate each updated triple with the updated sentence it was extracted from. As discussed previously, we count the number of occurrences for each such sentences. Variables $H_i^O$ and $H_i^U$ designates the number of occurrences for the original and updated sentence. Their value domains are non-negative integers. They are dependent on the truthfulness of the sentence whose occurrences they count (assuming that incorrect sentences are less likely to appear). Next, we introduce binary variables $C_i$, representing the class of the $i$-th update. We distinguish two classes: we distinguish updates introducing or removing factual mistakes from all other updates. The first category is interesting to us as we can extract factual mistakes. The class of update therefore depends only on the truthfulness of the two triple versions. Finally, we introduce variables $K_i$ to represent the set of keywords that appear in the updated sentence. The keywords in the sentence influence the expected number of Web hits (e.g., we would expect more hits for claims about movies than about birth dates of specific persons).

Variables $H_i$, $H_i^O$, $H_i^U$, and $K_i$ are observable. We mine the Web to count the number of occurrences for the original and updated sentence. Keywords are given by the update text. All other variables are latent. We are ultimately interested in finding out the value for variable $C_i$, representing the update class, for each update. To do so, we must link that variable to the observable ones. We start by analyzing the distribution of variables $H_i$, $H_i^O$, and $H_i^U$.

Variable $H_i$ is the number of Web hits for the $i$-th update. It is $H_i \sum_j X_j^i$ where the $X_j^i \in \{0,1\}$ are binary variables. Each variable $X_J$ indicates whether a Web hit for the $i$-th update was found in sentence number $j$. Summing over all sentences parsed on the Web yields the total number of Web hits. We treat the $X_j^i$ as independent, identically distributed Bernoulli variables. Hence, $\Pr(X_j^i = 1)$ is the probability that an average Web author expresses an opinion on the topic of the $i$-th update in any given sentence. As variables $X_j^i$ are independent and identical Bernoulli variables, the sum $\sum_j X_j^i$ follows a binomial distribution with parameters $n$ (the number of Web sentences) and $p_i$ (the probability that any given sentence refers to the $i$-th update):

$$\sum_j X_j^i \sim Bin(n, p_i) \tag{1}$$

The number of Web sentences ($n$) is large, the probability to find any specific topic in any given sentence is very small. This means that we can approximate the Binomial distribution by a Poisson distribution (counting the number of rare events over many tries). Hence, we obtain

$$\sum_j X_j^i \sim Poisson(\lambda_i) \tag{2}$$

where $\lambda_i = n \cdot p_i$ (the expected number of Web hits). The expected number of hits depends on the claim topic. Certain topics are more popular on the Web than others. We distinguish several popularity classes that are associated with different numbers for $\lambda_i$. Different popularity classes are associated with different keyword distributions. As discussed later, we learn to classify popularity based on keywords in an unsupervised approach.

For a given topic, it is possible to make accurate or inaccurate claims. We denote by $w_T$ the probability of finding accurate statements on the Web. Then $1 - w_T$ is the probability of finding an inaccurate Web statement. If $\lambda_i$ is the expected number of Web hits that address the $i$-th topic then $w_T \cdot \lambda_i$ is the expected number of hits for an accurate claim (using one specific formulation). Also, $(1 - w_T) \cdot \lambda_i$ is the expected number of hits for an inaccurate claim. The distribution of $H_i^O$ and $H_i^U$ depends therefore on the values for variables $T_i^O$ and $T_i^U$. We obtain

$$H_i^O \sim \begin{cases} Poisson(w_t \cdot \lambda_i) & \text{if } T_i^O = \textbf{true} \\ Poisson((1-w_t) \cdot \lambda_i) & \text{if } T_i^O = \textbf{false} \end{cases} \tag{3}$$

for $H_i^O$ (and analogously for $H_i^U$). Next, we analyze the distribution of the remaining random variables in our model. For $T_i^O$ and $T_i^U$ we assume an a-priori probability for correct and incorrect statements, i.e. $\Pr(T_i^O = \textbf{true}) = \Pr(T_i^U = \textbf{true}) = u_T$ (where $u_T \in [0,1]$ is the probability of correct updates). We use different parameters to represent truth probability for Web claims and Wikipedia updates. This seems required, especially since we consider a subset of Wikipedia updates that are more likely than other Wikipedia updates to correct factual mistakes.

**Example 6.1.** We illustrate semantics for some variables for our running example update (update 1). The correct assignment is of course $T_1^O = \textbf{false}$ and $T_1^U = \textbf{true}$ which means that this is a correcting update ($C_1 = \textbf{true}$). We cannot observe the values of those latent variables but must infer them. Assume $H_1^O = 3$ and $H_1^U = 9$. Given suitable parameter values representing the probability of Web and Wikipedia authors to issue truthful claims, we will infer the correct assignments for latent variables from the hits.

## 6.3 Expectation-Maximization

The goal of expectation-maximization is generally to maximize a likelihood function $L(\theta; \mathbf{X}, \mathbf{Z})$ where $\theta$ is a vector of (unknown) parameters, $\mathbf{X}$ observed and $\mathbf{Z}$ latent data. Here, $\theta = \langle w_T, u_T, \mathbf{T_M} \rangle$ includes Web and Wikipedia truth probabilities as well as parameters $\mathbf{T_M}$ of a Naive-Bayes model that classifies sentences into one out of several popularity categories, each associated with an expected number of Web hits. The observed data $\mathbf{X} = \langle h^O, h, h^U, K \rangle$ include Web hits ($h^O$, $h^U$, and $h$ for original, updated, and both versions) and sentence keywords $K$ (we denote the component for the $i$-th update via subscript notation). For the values of latent variables, we have $\mathbf{Z} = \langle t^O, t^U, c \rangle$ (i.e., the truth value for each original and updated sentence and the implied update type). The likelihood function is the product of all update-specific variable probabilities (which we assume to be independent): $L(\theta; \mathbf{X}, \mathbf{Z}) = \prod_i \Pr(\mathcal{H}_i \wedge \mathcal{T}_i | \theta, \mathbf{X}, \mathbf{Z})$ where $\mathcal{H}_i$ designates the event $H_i^O = h_i^O \wedge H_i^U = h_i^U \wedge H_i = h_i$ of obtaining the observed number of Web hits for the $i$-th update and $\mathcal{T}_i$ designates the event $T_i^O = t_i^O \wedge T_i^U = t_i^U$ of original and updated sentence having the latent truth values specified in $\mathbf{Z}$. Our goal is to find a combination of latent data (i.e., truth assignments for sentences) and unknown parameter values that maximize the likelihood function. To do so, we iteratively alternate between optimizing parameter values for fixed truth assignments and optimizing truth assignments for fixed parameters.

Before the first iteration, we assign tentative values to random variables according to a (simple) heuristic. For each Wikipedia update, we distinguish three cases. If we find more Web hits for the original sentence of the $i$-th update, compared to the updated one, we set $T_i^O = \textbf{true}$ and $T_i^U = \textbf{false}$. If we find more Web hits for the updated version, we set $T_i^O = \textbf{false}$ and $T_i^U = \textbf{true}$. If the number of Web hits is the same, we set $T_i^O = T_i^U = \textbf{true}$. Note that we make the default assumption that correct claims are more likely than incorrect claims in the absence of further information. This corresponds to commonsense knowledge and prior work in data cleaning often makes similar assumptions [22]. The initial value assignment is simplistic but we refine it in the following iterations. The initial values of other variables can be inferred from the value assignments for $T_i^O$ and $T_i^U$, as well as from the number of Web hits.

In each iteration, we first infer parameter values from the variable value assignments. Denoting by $m$ the total number of Wikipedia updates, we use

$$u_T = \frac{\sum_i (\mathbb{1}(T_i^O = \textbf{true}) + \mathbb{1}(T_i^U = \textbf{true}))}{2 \cdot m} \tag{4}$$

to estimate the probability of correct updates on Wikipedia. Also, we use the formula

$$w_T = \frac{\sum_i (h_i^O \cdot \mathbb{1}(T_i^O = \textbf{true}) + h_i^U \cdot \mathbb{1}(T_i^U = \textbf{true}))}{\sum_i (h_i^O + h_i^U)} \quad (5)$$

to estimate the probability of true claims on the Web (where $h_i^O$ and $h_i^U$ are the number of Web hits collected for original and updated sentence of the $i$-th update).

To infer the popularity of a topic (i.e., the expected number of Web statements that refer to it), we need to take into account the number of Web hits collected but also whether they refer to accurate or inaccurate claims about the topic. We only detect Web hits for sentences that appear in Wikipedia updates which may include only inaccurate or only accurate claims on the topic (in different formulations) or one accurate and one inaccurate claim. We estimate the popularity of a claim topic as

$$p_i = (\frac{h_i^O}{w_i^O} + \frac{h_i^U}{w_i^U})/2 \quad (6)$$

where $w_i^O = w_T \cdot \mathbb{1}(T_i^O = \cdot\textbf{true}) + (1-w_T) \cdot \mathbb{1}(T_i^O = \textbf{false}))$ is the truth-related "scaling factor" for the original sentence and $w_i^U$ the analogue scaling factor for the updated sentence.

We introduce several discrete "popularity classes" that are associated with an expected number of Web hits (powers of ten up to the maximal number of Web hits in our data set). We assume that popularity for a sentence can be roughly estimated based on keywords it contains. We train a Naive Bayes classifier to predict the popularity category of an update based on keywords that appear in original and updated sentence. Since popularity, as defined above, is a function of observed hits as well as latent truth assignments, we cannot train that classifier before applying expectation-maximization. Instead, we train the classifier anew in each iteration (its internal model parameters are part of the parameters to optimize, $\theta$).

In the second part of each iteration, we use the tentative parameter values to infer the most likely value for each variable. To do so, we simply use the formulas representing the distribution of each variable. We use the previously trained classifier to assign each update to a popularity category. For each Wikipedia update, there are four truth combinations (settings for binary variables $T_i^O$ and $T_i^U$) with regards to original and updated sentence. We calculate probability for each combination and assign the values with highest probability. We repeat iterations until convergence. We observed convergence (i.e., change in parameter values of less than 0.1%) already after 10 iterations. Based on the final result, we identify updates that correct a prior mistake. From those, we extract subject-predicate-object triples that represent mistakes and their corrections. The next section describes how we rank the resulting triples.

## 7. RANKING AKB ENTRIES

In the final step of the AKB mining pipeline, we rank extracted triples by their quality. Ideally, each extracted triple describes one factual mistake completely and concisely. A triple describes a mistake completely if it identifies subject, predicate, and object without ambiguity. A triple is concise if it does not provide additional information beyond the mistake it represents. Both, completeness and conciseness, are required to spot mistakes based on AKB triples. We rank triples based on their likelihood to satisfy both properties.

**Example 7.1.** The triple (book,released in,1978) is incomplete as it does not identify a specific book (and different books have different release dates). The triple (Einstein,published,special and general theory of relativity in 1905) is not concise as it states multiple facts (thereby making it unclear, which of them is erroneous).

Determining completeness and conciseness requires commonsense knowledge and deep natural language understanding. For instance, consider the triples (the book,was released in,1978) and (the book,was invented in,1450 along with the printing press). Here, the same subject is ambiguous for the first triple but not for the second. To maximize our coverage, we therefore rely on a set of simple and robust heuristics for ranking triples instead.

We associate each triple with a score. The higher the score, the higher the assumed triple quality. We calculate triple scores $S_t$ as the weighted sum of four terms:

$$S_t = w_s \cdot S_s + w_p \cdot S_p + w_o \cdot S_o + w_c \cdot S_c. \quad (7)$$

Here, $S_s$ is based on the triple subject, $S_p$ is based on the triple predicate, and $S_o$ is based on the object. $S_c$ is based on the complexity of the sentence structure and ranks simpler sentences higher. Next, we explain how to calculate those factors.

Our goal is to obtain triples with non-ambiguous subject. Pronouns (e.g., "It"), broad categories (e.g., "company"), or clauses (e.g., "The aforementioned items") may lead to ambiguous subjects. Intuitively, non-ambiguous subjects should appear less frequently (as ambiguous subjects accumulate occurrences over all entities they represent). Hence, we base our subject rank on the number of occurrences. Using the number of subject occurrences directly is insufficient. It fails for cases in which the subject is formed from a rare combination of unspecific words (e.g., subject "The company mentioned in the previous paragraph"). As a refinement, we consider occurrences frequencies $f_w$ of the words $W_s$ that subject $s$ is composed of:

$$S_s = 1/(1 + \log(\max_{w \in W_s}(f_w))) \quad (8)$$

This formula penalizes subjects that contain very frequent words. On the other side, occurrence frequency depends on many factors beyond ambiguity. We use the logarithm to lessen the chances of pruning out non-ambiguous subjects that are frequently mentioned. Using the maximum (instead of the average) of occurrence counts puts subjects with higher word counts at a disadvantage. This is intended as more subject text often hints at ambiguous paraphrases or superfluous information.

Ranking predicates is harder. We found that predicate occurrences frequency does not correlate well with ambiguity. The least frequent predicates are often due to extraction mistakes. Many highly specific and useful predicates (e.g., "be born in") contain frequent words. Hence, we judge predicates not based on the predicate text directly. Instead, we judge predicates based on the subjects they connect to.

For a fixed entity and predicate, the number of other, related entities is typically limited (many relationships such as "is capital of country" are even functional and map each entity to at most one other entity). Hence, having a predicate connecting the same entity to many other entities is atypical. We therefore want to decrease the rank of such predicates. However, we must distinguish between specific subjects and ambiguous subjects (e.g., pronouns or broad categories). Predicates should not be penalized for appearing in many triples with pronouns (such triples will however receive a low subject score). Hence, we must also consider the likelihood that subjects are specific which connects to their occurrence frequency.

We calculate the score of a predicate $p$ based on the set of triples $T_p$ in which it appears, the occurrence frequency $f_s^p$ of subject $s$ in triples with predicate $p$, and the general occurrence frequency $f_s$ of subject $s$:

$$S_p = 1 - (\sum_{t \in T_p} (f_s^p / f_s)) / |T_p| \quad (9)$$

The formula above does not take into account the number of times that a predicate is used in general. Applying it without further constraints tends to benefit low-quality predicates that appear only very infrequently (but with highly distinctive subjects). We therefore apply the formula above to predicates whose number of triples is above a threshold (while assigning a low standard score to the remaining ones).

Finally, we integrate the triple object. In our experience, the primary problem with regards to triple objects is conciseness. Triple objects often contain superfluous information after extraction. This can create ambiguity with regards to which pieces of information are incorrect. We use again the length of the object text as a proxy for the likelihood that superfluous information is conveyed. Denoting text length of object $o$ by $|o|$, we use the following formula:

$$S_o = 1/(1 + \lceil \log_{10}(1 + |o|) \rceil) \quad (10)$$

Using the logarithm (with base 10) reduces again the impact of this criterion. We additionally round logarithm results. This ensures that for objects with similar length the other two factors (related to subject and predicate) take precedence. This is in line with our observation that triple objects tend to have the lowest impact on triple quality.

The complexity score, $S_c$, is based on the complexity of the sentence parse trees. We penalize high complexity as it increases the probability of extractor mistakes. Our metric is based on the number of edges and the tree depth of the dependency tree. Complexity is $S_c = 1 - 1/(\delta^2 + 1)$ where $\delta$ is the deviation from the average according to those metrics.

The sum over all four factors (subject, predicate, object, and complexity score) yields a simple but effective ranking criterion. We clearly trade recall for precision by penalizing for instance triples with very common subjects. Still, if the goal is to obtain a reduced number of high-quality triples, the given heuristic is a useful tool. We analyze its precision-recall tradeoffs in more detail in Section 8.

## 8. EXPERIMENTAL RESULTS

We applied the pipeline described before to Wikipedia updates and correlated updates with a snapshot of the entire Web. We mined over 100,000 ranked factual mistakes with a precision of up to around 80% (first tier). In this section, we report generation statistics such as processing time and input and output sizes for different pipeline stages (see Section 8.1). We compare against baselines and assess pipeline steps (see Section 8.2), evaluate (via manual verification) the "precision" of mined mistakes (see Section 8.3) and derive several first insights from the AKB data (see Section 8.4). Finally, we show that the mined data can be used to find new mistakes on the Web (see Section 8.5).

**Table 1: Breakdown of processing time on cluster with 2,500 machines.**

| Task | Processing Time |
|---|---|
| Extracting entity swaps | 10h |
| Extracting number swaps | 5h45 |
| Counting entity-swap Web hits | 1h4 |
| Counting number-swap Web hits | 2h12 |
| Matching AKB entries to Web | 8h |

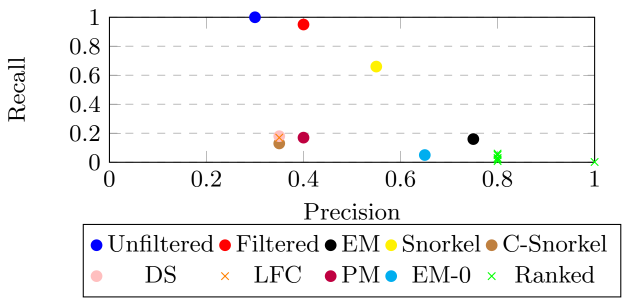

**Figure 3: Sentence precision and recall of different approaches for entity-swap updates.**

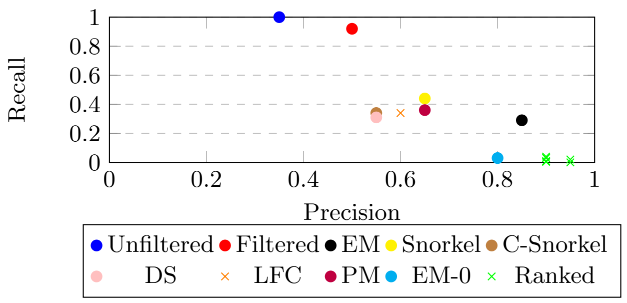

**Figure 4: Sentence precision and recall of different approaches for number-swap updates.**

### 8.1 Pipeline Statistics

**Table 2: Sizes of intermediate results.**

| Data Set | Size | Card. |
|---|---|---|
| Wikipedia Updates (raw input) | 16 TB | 39.65M |
| Entity swaps | 700 MB | 1.7M |
| Number swaps | 600 MB | 2.9M |
| Entity swap AKB | 8 MB | 41K |
| Number swap AKB | 14 MB | 76K |

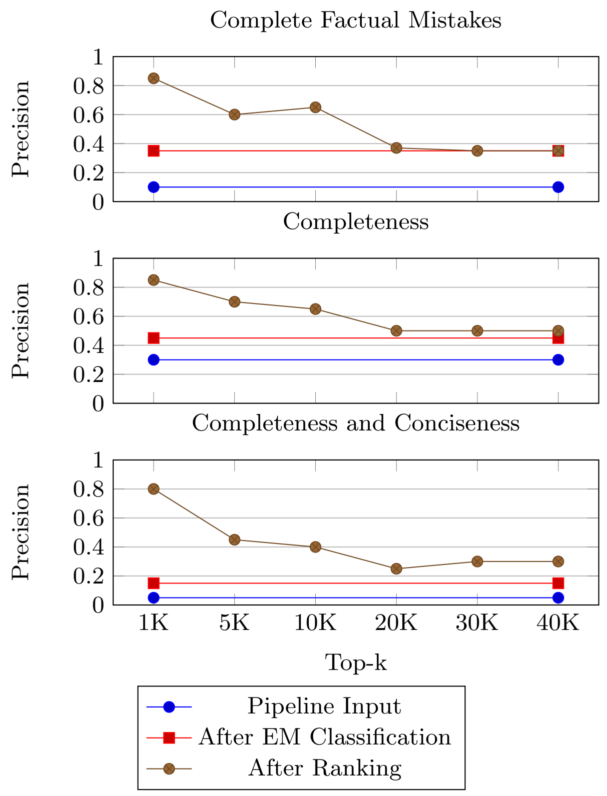

**Figure 5: Triple precision of AKB single entity-swap entries as a function of rank.**

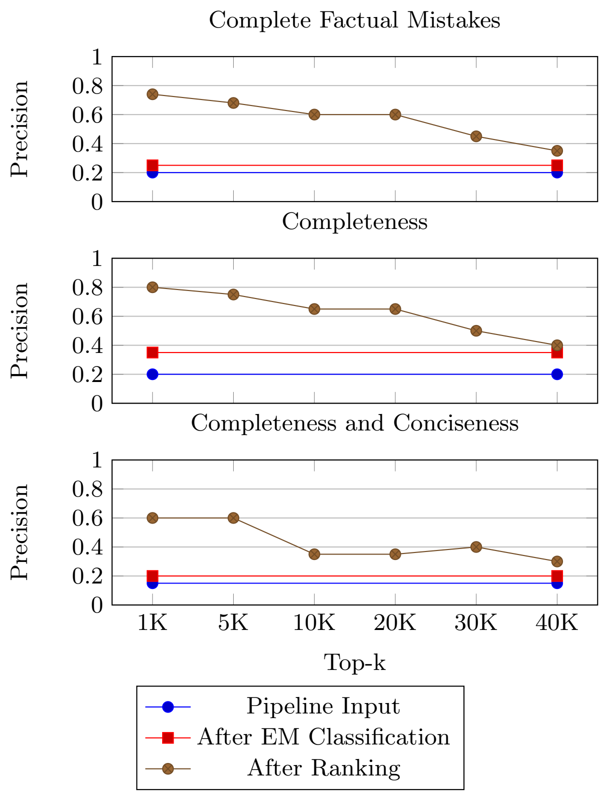

**Figure 6: Triple precision of AKB number-swap entries as a function of rank.**

We processed all English Wikipedia pages along with the most recent 1000 revisions, up to August 10, 2018. This amounts to an input size of 16 terabytes, describing 39.65 Wikipedia updates in total. We executed the pipeline on a large cluster with 2,500 nodes. Running the pipeline on the full input took around 27 hours.

Table 1 shows processing time for different steps. Extracting entity swaps and number swaps from raw Wikipedia updates accounts for a combined 16 hours of processing time. The time required for extracting interesting entity swaps is higher than the time required for number swaps. This is mostly due to the computational overhead of entity resolution (for matching entities in text to entries in the Google knowledge graph). This overhead is specific to the extraction of entity swaps. To apply the probabilistic model, we need to count the number of references to original and up-

**Table 3: Breakdown of entity swap updates.**

| Update Type | Percentage |
|---|---|
| Controversial updates | 20% |
| Substring containment | 18.7% |
| Likely spelling mistake | 10% |
| Geographic containment | 6.7% |
| Synonym updates | 2.6% |
| Correction candidates | 42% |

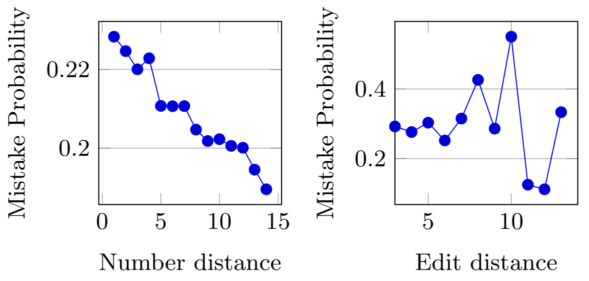

**Figure 7: Correlating the probability that Web authors confuse numbers or entities with numerical and edit distance.**

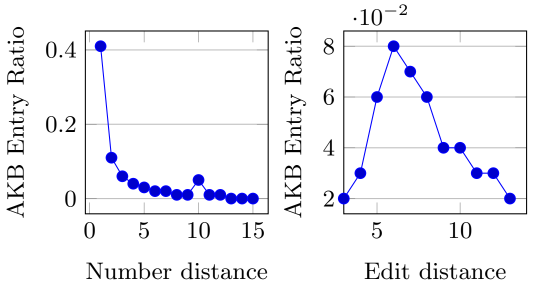

**Figure 8: Correlating the ratio of AKB entries with numerical or edit distance between swapped entities or numbers.**

dated sentence. The time required for counting dominates time for inference. Counting references takes over three hours in total. As we extract more number swaps (see next), the time required for counting number swap references is slightly higher. As discussed later in more detail, we also matched statements on the Web against AKB entries. This step took eight hours of processing time.

Table 2 specifies input and output sizes for different pipeline stages. Clearly, the first stage of the pipeline is the most selective one. It reduces input data by a factor of eight. The heuristics of the first stage are therefore quite effective at reducing overheads for the following stages. The input data sizes correlate with processing overheads (discussed before). Classification further reduces data by a factor of four. In total, we extract around 130,000 ranked AKB entries.

Specifically for entity swaps, we apply various heuristics in the first stage to discard irrelevant updates. Table 3 shows a detailed breakdown of entity swap updates. Only 42% of entity swap updates pass all heuristic tests and are used as input to the probabilistic model. Controversial updates account for the largest percentage (20%) of other updates. This is not necessarily due to the large number of controversial topics. According to our observations, there are few controversial topics which motivate Wikipedia authors to mutually revert their updates. We observed such cases in particular for religious topics and topics related to territorial conflicts. Specializations are common types of updates as well and our simple heuristics of substring containment is already able to discard a large number of cases. Table 8.1 shows a few examples of categories assigned by those heuristics.

## 8.2 Comparison versus Baselines

We compare our approach against several baselines. Also, we evaluate output quality achieved after different pipeline stages. The results are reported in Figures 3 (entity-swap updates) and 4 (number-swap updates). We use precision and recall as criteria. Precision is evaluated by manually verifying for sentences, marked up as erroneous by our pipeline or baselines, contain factual mistakes indeed. We consider a factual mistake as verified if it contradicts information extracted from an authoritative source (e.g., a company's Web site for a claim about that company's founding date). If we cannot identify a single authoritative source, we correlate information of different types from multiple sources (e.g., exploiting birth year and years at which degrees were obtained to verify age). We evaluated 20 randomly selected output sentences for each data point. We report the percentage of actual mistake sentences as precision. As recall, we use the estimated number of factual mistakes in the output of a method (obtained by multiplying the precision of the method by its output size), scaled to the estimated total number of mistakes in raw input (obtained by multiplying the fraction of mistakes in a random sample of 20 updates from the raw input by the total number of updates).

We compare against Snorkel [20], a generic system for weakly supervised learning. Snorkel uses the results of our heuristics, described in Section 5, as well as the number of Web hits ($h = \langle h_O, h_U \rangle$) as inputs. Snorkel dominates several of the other baselines but does not reach the precision of our specialized approach. Our goal is to infer truth from noisy crowd input (i.e., Wikipedia and Web authors) in a specific scenario. Hence, we also compare against three generic algorithms that infer correct answers from noisy crowd input: PM [2,15], D&S [4], and LFC [21]. Those algorithms process more fine-grained input, compared to our model: they integrate not only the number of votes for different answer options (i.e., the number of Web hits in our scenario) but also model the specific voters (i.e., the URLs of Web hits in our scenario). This allows them to infer voter-specific reliability values. We therefore model the URLs of Web hits as voters and original and updated sentence for each update as answer options. Beyond the three aforementioned algorithms, we also use Snorkel with URLs as labeling functions ("C-Snorkel" in the plots). In all cases, the resulting precision-recall tradeoff is sub-optimal.

The plots show the precision-recall tradeoffs realized by the output of different pipeline stages: before filtering ("Unfiltered"), after filtering ("Filtered"), after applying the probabilistic model based on expectation-maximization ("EM"), and for the top-1 to 40K triples after ranking ("Ranked"). As intended, each pipeline stage gradually trades recall for precision. Note in particular that filtering reduces recall only slightly compared to the reduction in data size. This means that the number of false negatives of our heuristics is low. Applying the probabilistic model leads to the largest increase in precision. To test whether parameter learning via expectation-maximization is indeed helpful, we also tested a version that avoids any expectation-maximization iterations and uses instead the labels assigned before the first iteration (EM-0) before ranking. Clearly, precision is significantly worse without optimized model parameters.

## 8.3 Evaluating Triple Quality

In the last subsection, we calculated precision at the sentence level (i.e., the fraction of sentences containing mistakes). Doing so was required to allow a fair comparison against baselines, some of which use input data that is only available at the sentence level (e.g., the number of Web hits is only available per sentence, not per triple, and forms the input for algorithms D&S, LFC, and PM). In this subsection, we evaluate the final output of our baseline, ranked triples, according to more fine-grained metrics. Ideally, each AKB entry has the following properties. First, it describes a claim completely without ambiguity (completeness). Second, the claim is actually a factual mistake. Third, the entry does not contain any claims beyond the factual mistake (conciseness). We evaluate triple precision based on the subset of desirable properties that they satisfy. The metrics that we use in this subsection are therefore *stricter* than the ones we used before. Hence, triple precision, used here, is generally lower than sentence precision, used before.

Figure 5 shows results for factual mistakes mined from entity-swap updates (i.e., replacing one entity by another one). We report precision as the percentage of completely described factual mistakes (our primary metric), as the percentage of completely described claims, and as the percentage of complete and concise entries. Our final output is ranked and we report precision for entries selected randomly from the top-$k$ entries (for different values of $k$, represented on the x axis). We selected 20 entries for evaluation for each $k$. Besides final output, we also report precision for the raw input (filtered Wikipedia updates) and for the results after applying the EM stage. Those two intermediate results are not yet ranked (hence, the results are represented as horizontal lines in Figure 5).

Table 4: **Examples of Wikipedia sentences with their categories assigned by first pipeline stage.**

| Wikipedia Sentence | Swap | Category |
|---|---|---|
| He remains one of the most recognisable figures in modern English music . | English → British | Synonym |
| He was nominated by President Bill Clinton on January 27 , 1998 , during Clinton 's second term , and was confirmed by the Senate on May 5 , 1998 . | Senate → United States Senate | Sub-string (specialization) |
| Londonderry Air is an air that originated from County Londonderry in Ireland. | Ireland → Northern Ireland | Sub-string (specialization) |
| John Graham McVie was born on November 26th , 1945 , in Ealing , West London , England to Reg and Dorothy McVie and attended Walpole Grammar School . | England → United Kingdom | Geo-contained (generalization) |
| The Portugese monarchy was abolished on the 5th October 1910 when King Manuel II was deposed following a republican revolution . | Portugese → Portuguese | Spelling Error |
| It was ready for consecration in the spring of 516 BC , more than twenty years after the return from captivity . | BC → BCE | Controversial |
| Aqua is the Greek word for water . | Greek → Latin | Factual Mistake |
| The official language of Brazil is French . | French → Portuguese | Factual Mistake |
| In mid- 2004 , Oracle closed their Newark , California , factory and consolidated all manufacturing to Hillsboro , Oregon . | Oracle → Sun | Factual Mistake |

Without further processing, the percentage of completely described factual mistakes is very low (10%). Applying expectation-maximization classification increases that precision to 35%. Triple ranking finally boosts precision to up to 80% for the first thousand entries (72% for the first twenty thousand entries). We observe similar tendencies according to the other two metrics. Clearly, each pipeline steps leads to a significant increase of the quality. Also, the ranking approach is effective in putting entries of higher quality to the front.

Figure 6 shows analogue results for entries mined from number swap updates. We observe a precision of 74% within the first thousand entries (and 60% within the first twenty thousand entries). The relative tendencies between the results after different pipeline stages are the same as before.

## 8.4 AKB Insights

We demonstrate that we can test hypothesis and derive new insights from the AKB. Such insights are generally interesting and may inform the design of future fact checking tools. Analysis results come with a caveat since our data source and extraction methodology may introduce bias. Still, the AKB can be used as one of several sources of evidence and AKB results can guide the design of more targeted studies. We start by verifying a simple hypothesis:

**Hypothesis 8.1.** The probability to confuse two numbers or entities increases if they are similar.

We measure the similarity between numbers by the absolute distance between them. We measure the similarity between entities by the edit distance. We use two indicators for the probability to make a mistake. We consider the ratio of AKB entries for a fixed (numerical or edit) distance. Also, for a correcting update with $h_O$ Web hits for the original (incorrect) version and $h_U$ hits for the updated (correct)

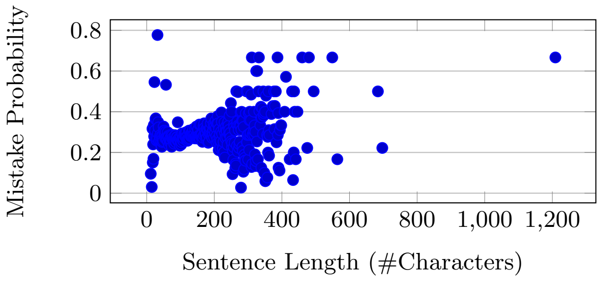

**Figure 9: Mistake probability versus sentence length.**

version, we use $h_O/(h_O + h_U)$ as a proxy for the probability that an average Web author makes the mistake.

Figure 7 shows the dependency between similarity and the probability that Web authors confuse two numbers or entities. In particular for number swaps, there is a clear correlation. The probability of confusing two numbers decreases with the distance between those numbers. For the edit distance, the tendency is less clear. Note that we report results only for entity swaps with an edit distance of at least three (since we filter out swaps with a lower edit distance).

Figure 8 shows the ratio of AKB entries as a function of numerical or edit distance. Again, the ratio of AKB entries is higher for cases in which close numbers are confused. Staggering 40% of numerical AKB entries concern cases in which two numbers with a distance of one were confused (e.g., people commonly confuse the year of an event with the preceding or following one). The correlation is less clear for the edit distance.

**Table 5: Examples for Web matches generated by baseline using large knowledge base.**

| Sentence | Triple | Paths found by KG | Category |
|---|---|---|---|
| Peter Williams is synonymous with UK fast fashion . | {Peter Williams, national-affiliation, UK} | {Peter Williams, national affiliation, Jamaica} | Extractor Error |
| Gloria Steinem and Dorothy Pitman Hughes - 1971 Dorothy Pitman Hughes is a writer , speaker , activist and a lifelong champion for women , children and families . | {Dorothy Pitman Hughes, profession, writer} | {Dorothy Pitman Hughes, profession, Author/Orator/Advocate} | KG Incomplete |
| Brenntag recently agreed to acquire the assets of Lubrication Services LLC ( LSi ) . | {Lubrication Services LLC, parent organization, Brenntag} | {Lubrication Services LLC, parent organization, Coastal Chemical Co., LLC/Enterprise GP Holdings/TEPPCO Partners} | KG not up-to-date |

**Table 6: Ratio of occurrences in AKB versus occurrences in raw input for triples with certain predicates.**

| Predicate | AKB Boost |
|---|---|
| be bear in | 5.7 |
| direct by | 5 |
| be bear on | 4.8 |
| star | 4 |
| graduate in | 4 |
| open in | 4 |
| be hold on | 4 |
| move in | 4 |
| form in | 4 |
| be found in | 3.7 |

In summary, we find clear evidence that people are more likely to confuse similar numbers. This insight can be directly used to improve existing fact checking tools. For instance, a recently proposed tool assesses the probability that natural language statements about numerical properties are incorrect [13]. Our findings could be used to refine the probability model by taking into account the distance between the number claimed in text and the value the system believes to be correct.

It is less clear whether people are likely to confuse entities with a small edit distance. Perhaps, a different metric of similarity (e.g., based on entity type or semantic similarity) between entities must be used. We leave further analysis to future work. Now, we verify a second hypothesis:

**Hypothesis 8.2.** The probability to make a mistake depends on the relationship that a claim refers to.

We collect evidence for this hypothesis as follows. We consider the ratio of triples with certain predicate extracted from the raw Wikipedia update stream. Then, we compare that ratio against the ratio of triples with that predicate in the AKB. We verify whether there are certain predicates that appear over-proportionally often in the AKB.

Table 6 shows the 10 predicates for which the number of occurrences increased by the highest factor, comparing raw input and AKB result. The higher that ratio (called "AKB Boost" in the table), the higher the probability to make mistakes. Clearly, certain predicate are over-represented in the AKB. This provides supporting evidence for our hypothesis. For instance, the ratio of triples for the predicate "be bear in" (associating a person with birth year or place) increases by nearly factor six. Such insights can be useful to set a-priori probabilities for erroneous claims in verification, based on predicates.

Finally, we consider the following hypothesis:

**Hypothesis 8.3.** The probability to make mistakes increases when writing longer sentences.

To test this hypothesis, we calculate mistake probability of Web users (as described before) as a function of sentence length (measured as the number of characters). Figure 9 shows the results. Clearly, we find not strong evidence to support our hypothesis. We cannot confirm that Web authors are more likely to make mistakes when writing longer sentences.

## 8.5 Application: Fact-Checking the Web

We demonstrate a first use case for the AKB. We show that the mistakes we mine can be used to fact check other Web sites beyond Wikipedia. We thereby verify our hypothesis that factual mistakes tend to be made more than once. Also, we show that naive approaches to exploit knowledge bases for identifying factual mistakes on the Web fail.

Our goal is to find factual mistakes on the Web. We compare two approaches. The first approach exploits a sample from our AKB (we use the first thousand or 20K samples from the number swaps AKB). We match AKB entries against subject-predicate-object triples extracted from the entire Web. To do so, we mine a snapshot containing 2.3 billion Web sentences and apply information extraction methods. We match the extracted triples against AKB entries. We consider Web sentences as erroneous *if they match* one of the AKB entries. To increase efficiency, we only extract triples from sentences that contain subject and object string from one of the AKB entries. Still, matching the entire Web

**Table 7: Examples for Web matches generated for AKB entries.**

| Wikipedia Sentence | Correction | Web Sentence |
|---|---|---|
| Mr. Khera is the author of 12 books including his first book You Can Win ( Jeet Aapki in Hindi ) was published in 1998 | 12 to 16 | Mr. Khera is the author of 12 books including international bestseller You Can Win , which has sold over 2 million copies in 16 languages. |
| The P8 was adopted as the standard issue handgun for the Bundeswehr in 1994 and the G36 was adopted in 1997. | 1997 to 1995 | HK felt their HK50 project , in development since the mid-1970s was a better bet than the heavier G41 , and following Bundeswehr trials the G36 was subsequently adopted in 1997. |
| The subgenus Glycine consists of at least 16 wild perennial species : for example , Glycine canescens F.J. Herm. | 16 to 25 | At present , the subgenus Glycine consists of at least 16 wild perennial species : for example , Glycine canescens , and G. tomentella Hayata found in Australia , and Papua New Guinea. |
| Buland Darwaza , also known as the Gate of Magnificence , was built by Akbar in 1601 A.D. at Fatehpur Sikri. | 1601 to 1576 | Buland Darwaza was built by the great Mughal Emperor Akbar in 1601 A.D. |
| The first recorded use of the present-day kanji ( characters ) , which mean " carrying rice , " ( literally is " rice load " ) was in the Ruij Kokushi in 827 A.D. Other sets of kanji with the same phonetic readings , most of which contained a reference to rice , were in use earlier , and most scholars agree that the name Inari is derived from. | 827 to 892 | The first recorded use of the present - day kanji ( characters ) , which mean " carrying rice , " was in the Ruij Kokushi in 827 A.D. |

**Table 8: Identifying mistakes on the entire Web: comparing AKB and KB based approach.**

| Approach | #Matches | Precision |
|---|---|---|
| AKB 1K: Match is Mistake | 330 | 75% |
| AKB 20K: Match is Mistake | 6147 | 50% |
| KB Miss is Mistake (Sample) | 469 | 0% |

against an AKB extract took around eight hours on a cluster with 2,500 nodes.

For comparison, we exploit the Google knowledge graph for fact checking via a simple approach. We exploit text analysis tools that extract triples from Web sentences and match subject, predicate, and objects to entities and relations in the Google knowledge graph. We consider sentences as incorrect if extracted triples *do not match* entries in the knowledge base. This approach classifies a large number of Web sentences as likely mistakes. We therefore use a small sample of only 0.0002% of the entire Web containing $324,580$ sentences. We extracted $2,253$ triples that we were able to fully match to KB entries (i.e., we need to resolve subject, predicate, and object). Note that the number of extracted triples would be much higher without that requirement. Out of those, 469 triples were classified as likely mistakes (since they do not correspond to a registered relationship).

We assess the precision of the two approaches. For each approach, we select 20 random entries from the result set (i.e., Web sentences that were classified as likely mistakes). We manually verify whether the sentences contain actual mistakes. Table 8 summarizes the result. Clearly, considering the absence of KB entries as evidence for factual mistakes is impractical. in three out of ten cases, false positives were due to the KB being incomplete. In seven out of ten cases, a mistake of the extractor prevents a KB match. The number of likely mistakes returned by the AKB-based approach is much lower. However, 70% of returned matches are actual mistakes. Three false positives are due to wrong AKB entries (i.e., entries that are no actual mistakes). The other three are due to ambiguous AKB entries (i.e., the AKB entry was a mistake in its original context but the extracted triple is too generic).

Table 8 shows a few examples for Web matches, generated from AKB entries. It shows the updated sentence on Wikipedia, from which the AKB entry was extracted. It also shows the Web sentence in which the same mistake appears.

## 9. CONCLUSION AND OUTLOOK

We described our efforts to build an anti-knowledge base from Wikipedia updates. We describe a three-stage pipeline that uses raw updates as input and produces ranked anti-knowledge triples as output. We produced over 110,000 ranked triples from 16 terabytes of input data, exploiting over 24 hours of computation time on a large cluster.

We presented first evidence that the mined data can be useful. In particular, we verified several hypothesis around why users make mistakes. We also exploited the mined knowledge for fact checking other parts of the Web, successfully identifying factual mistakes with high precision. This demonstrates that the mined mistakes are not uncommon and can be used for fact checking.

## 10. REFERENCES

[1] G. Angeli, M. J. Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *ACL*, pages 344 –354, 2015.

[2] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[3] A. Bonifati, W. Martens, and T. Timm. Navigating the maze of wikidata query logs. In *WWW*, pages 127–138, 2019.

[4] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[5] C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré. Rapidly mixing gibbs sampling for a class of factor graphs using hierarchy width. In *NIPS*, pages 3097–3105, 2015.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110, 2004.

[8] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[9] FullFact. The state of automated factchecking. 2016.

[10] N. Hassan, B. Adair, J. T. Hamilton, C. Li, M. Tremayne, J. Yang, and C. Yu. The quest to automate fact-checking. In *Proceedings of the 2015 Computation+Journalism Symposium*, 2015.

[11] N. Hassan, F. Arslan, C. Li, and M. Tremayne. Toward automated fact-checking: detecting check-worthy factual claims by claimbuster. In *KDD*, pages 1803–1812, 2017.

[12] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, et al. Claimbuster: the first-ever end-to-end fact-checking system. *VLDB*, 10(12):1945–1948, 2017.

[13] S. Jo, I. Trummer, W. Yu, X. Wang, C. Yu, D. Liu, and N. Mehta. Verifying text summaries of relational data sets. In *SIGMOD*, pages 299–316, 2018.

[14] L. Li, H. Deng, A. Dong, Y. Chang, R. Baeza-Yates, and H. Zha. Exploring query auto-completion and click logs for contextual-aware web search and query suggestion. In *WWW*, pages 539–548, 2017.

[15] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198. ACM, 2014.

[16] J. Lichtarge, C. Alberti, S. Kumar, N. Shazeer, N. Parmar, and S. Tong. Corpora generation for grammatical error correction. In *NAACL*, pages 3291–3301, 2019.

[17] X. Lin and L. Chen. Domain-aware multi-truth discovery from conflicting sources. *VLDB*, 11(5):635–647, 2018.

[18] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

[19] R. Qian. Understand Your World with Bing. https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/, 2013.

[20] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *VLDB*, 11(3):269–282, 2017.

[21] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11(Apr):1297–1322, 2010.

[22] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *VLDB*, 10(11):1190–1201, 2017.

[23] H. Roitman, S. Hummel, E. Rabinovich, B. Sznajder, N. Slonim, and E. Aharoni. On the retrieval of wikipedia articles containing claims on controversial topics. In *WWW*, pages 991–996, 2016.

[24] schema.org. ClaimReview. https://schema.org/ClaimReview, 2016.

[25] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. *VLDB*, pages 1033–1044, 2007.

[26] A. Singhal. Introducing the Knowledge Graph: things, not strings. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, 2012.

[27] F. M. Suchanek, M. Sozio, and G. Weikum. Sofie: A self-organizing framework for information extraction. In *WWW*, pages 631–640, 2009.

[28] I. Trummer, A. Halevy, H. Lee, S. Sarawagi, and R. Gupta. Mining subjective properties on the Web. In *SIGMOD*, pages 1745–1760, 2015.

[29] X. Wang, C. Yu, S. Baumgartner, and F. Korn. Relevant document discovery for fact-checking articles. In *WWW, Journalism, Misinformation, Fact Checking Track*, pages 525–533, 2018.

[30] D. Yang, A. Halfaker, R. Kraut, and E. Hovy. Identifying semantic edit intentions from revisions in wikipedia. In *EMNLP*, pages 2000–2010, 2017.

[31] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *KDD*, pages 796–808, 2007.

[32] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *VLDB*, 10(5):541–552, 2017.

[33] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, pages 101–110, 2009.