Hierarchical Trivia Fact Extraction from Wikipedia Articles

Jingun Kwon[1], **Hidetaka Kamigaito**[1], **Young-In Song**[2] **and Manabu Okumura**[1]
[1]Tokyo Institute of Technology
[2]Naver Corporation
kwon.j.ad@m.titech.ac.jp
{kamigaito,oku}@lr.pi.titech.ac.jp
song.youngin@navercorp.com

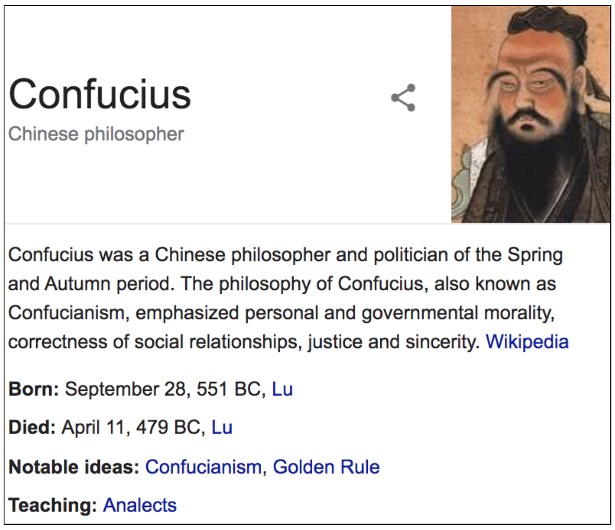
Abstract

Recently, automatic trivia fact extraction has attracted much research interest. Modern search engines have begun to provide trivia facts as the information for entities because they can motivate more user engagement. In this paper, we propose a new unsupervised algorithm that automatically mines trivia facts for a given entity. Unlike previous studies, the proposed algorithm targets at a single Wikipedia article and leverages its hierarchical structure via top-down processing. Thus, the proposed algorithm offers two distinctive advantages: it does not incur high computation time, and it provides a domain-independent approach for extracting trivia facts. Experimental results demonstrate that the proposed algorithm is over 100 times faster than the existing method which considers Wikipedia categories. Human evaluation demonstrates that the proposed algorithm can mine better trivia facts regardless of the target entity domain and outperforms the existing methods.

1 Introduction

Modern search engines offer a rich knowledge panel that appears on the right side of a user screen as a search result for a given entity to improve user engagement; See an example in Figure 1. One piece of information presented in the knowledge panel can be a trivia fact about the given entity, which contributes to the effective user engagement (Tsurel et al., 2017). For example, the Google search engine provides a trivia fact for a given entity to attract users' attention (Korn et al., 2019). Successfully attracting users' attention can facilitate the users to revisit the search engine (O'Brien and Toms, 2008). In contrast, poor user engagement would result in the users' switching to a competing search engine (White and Dumais, 2009). Trivia facts are, therefore, intended to improve the search quality to increase user engagement by making the search process more interesting.

In contrast to common, expected, or normal information, a trivia fact is an interesting fact that is unusual, unexpected, or unique (Prakash et al., 2015). Tsurel et al. (2017) demonstrated that trivia facts are usually little-known facts, thus, well-known facts (e.g., basic background information) about a given entity cannot be considered as trivia. For example, taking Confucius as an example, the following fact in the Wikipedia article can be considered as trivia: *"There is a grand-scale memorial ceremony, called Seokjeon Daeje, twice a year in South Korea for Confucius."* It is unusual and unexpected for such a memorial ceremony to be held in South Korea because Confucius is a popular Chinese philosopher and politician. In addition, surprising facts can be also considered as trivia because surprise and trivia worthiness are strongly correlated (Tsurel et al., 2017). For example, the sentence in the Wikipedia article for the film 'The Dark Knight Rises' states: *"On July 20, 2012, during a midnight showing of The Dark Knight Rises at the Century 16 cinema in Aurora, Colorado, a gunman wearing a gas mask opened fire inside the theater, killing 12 people and injuring 58 others."* Such information is surprising and can be considered as trivia. Those little-known and surprising facts from Wikipedia articles can be used to attract users' attention when searching these entities.


Confucius
Chinese philosopher

Confucius was a Chinese philosopher and politician of the Spring and Autumn period. The philosophy of Confucius, also known as Confucianism, emphasized personal and governmental morality, correctness of social relationships, justice and sincerity. Wikipedia

Born: September 28, 551 BC, Lu

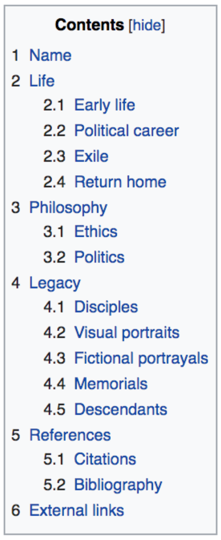
Died: April 11, 479 BC, Lu

Notable ideas: Confucianism, Golden Rule

Teaching: Analects

Figure 1: An example Google knowledge panel for the Chinese philosopher Confucius.

Early efforts to mine trivia facts took advantage of relational databases for relational expressions using several functions such as relational algebra, in order to generate trivia questions (Merzbacher, 2002). However, this approach requires human experts to generate a natural trivia question from obtained relational expressions. Rather than using a relational database, Lin and Chalupsky (2003) used the notion of "rarity" to measure interestingness to discover interesting facts and hidden connections such as relationships between people. Recently, Prakash et al. (2015) demonstrated that a Wikipedia article for a given entity can be a good source to mine trivia facts. In response to this report, there have been several attempts to develop automatic trivia fact extraction algorithms for various Wikipedia article domains such as movies (Prakash et al., 2015), TV shows (Niina and Shimada, 2018), and people (Tsurel et al., 2017). In addition, DBpedia, which provides structured data for Wikipedia contents, can be used to mine trivia facts for the domains of artists and actors (Fatma et al., 2017). Despite the success of the previous studies, however, the existing trivia fact mining algorithms face the following two issues: they are strongly dependent on the target domains and incur high computational cost. To address these issues, we aim to develop a new automatic trivia fact extraction algorithm that focuses on the Wikipedia's hierarchical structure.


Contents [hide]

1 Name
2 Life
  2.1 Early life
  2.2 Political career
  2.3 Exile
  2.4 Return home
3 Philosophy
  3.1 Ethics
  3.2 Politics
4 Legacy
  4.1 Disciples
  4.2 Visual portraits
  4.3 Fictional portrayals
  4.4 Memorials
  4.5 Descendants
5 References
  5.1 Citations
  5.2 Bibliography
6 External links

Figure 2: The hierarchical structure of the Confucius's Wikipedia article.

The structure in a Wikipedia article has not been necessarily considered, even though each article shares a standard format for the hierarchical structure including its article summary. Each Wikipedia article begins with its summary, which allows users to capture information about the given entity at a glance. Essentially, the summary presents the principal subjects by omitting less important content in the article. Then, the Wikipedia article is divided by content, following the summary. Figure 2 shows an example hierarchical structure of the Confucius's article. The text in his Wikipedia article is hierarchically divided, starting from a section, a subsection, a paragraph to a sentence. While the second section "Life" contains four subsections, the first section "Name" does not contain subsections.

Therefore, in this paper, we exploit the Wikipedia's shared format, the article summary and the hierarchical structure. We propose the *“Hierarchical Trivia Miner (HTM)”* algorithm, which extracts trivia facts in a top-down manner with a surprise score in a given article. The HTM algorithm uses a single Wikipedia article for a target entity and adopts a top-down approach; thus, it can mine trivia facts regardless of target domains and does not need high computation time because it gradually narrows down its search space for extracting trivia facts.

Human evaluation shows that our HTM algorithm can extract trivia facts from Wikipedia articles regardless of target domains. We also compare its performance of extracting trivia facts and the computation time with the previous method that makes use of Wikipedia’s category information.

2 Related Work

Recently, significant progresses have been made for mining trivia facts, and Wikipedia is currently widely used as a source to mine trivia facts. Existing approaches can be divided into two types, extracting trivia sentences and generating trivia sentences.

To extract trivia facts, Prakash et al. (2015) employed the ranking support vector machine (SVM). They utilized user-generated trivia from the Internet Movie Database (IMDb) as the training dataset and applied the trained model to a sentence in an entity’s Wikipedia article as a trivia candidate. Thus, it selects trivia sentences from the target Wikipedia article. However, the IMDb dataset handles only the movie domain, and it would be very costly to create datasets for other domains. Fatma et al. (2017) focused on the domains of Hollywood actors and music artists to harvest trivia facts. They classified trivia facts from the DBpedia dataset using fusion-based convolutional neural networks by incorporating hand-crafted features of a given fact triple, such as an entity, a relation/predicate, and an object.

Trivia facts can also be generated using Wikipedia articles. Tsurel et al. (2017) used the Wikipedia’s category information located at the bottom of articles. Their unsupervised algorithm ranks each category name based on “Surprise” and “Cohesiveness” scores and generates trivia facts using a template, “A is a member of C”, where “A” is a given entity and “C” is the selected category name. They used many articles from all categories of the target entity; thus, their algorithm incurs high computational cost. In addition, their algorithm only targets at the people domain because other domains do not necessarily contain trivia categories. Korn et al. (2019) focused on Wikipedia’s superlative tables as a natural source of interesting facts. The rows in the tables are sorted as the ranking of entities, based on the corresponding value such as the building’s height. Given the ranked list in the table, they tracked an interesting value and paired it with a candidate template to generate trivia facts.

Unlike previous methods, the proposed HTM algorithm extracts trivia facts for a given entity, using a single Wikipedia article based on its hierarchical structure and summary. Using hierarchical information is currently widely applied in various tasks as a good indicator to improve the task performance, such as text classification (Yang et al., 2016), text summarization (Christensen et al., 2014), language modeling (Gulordava et al., 2018), and neural machine translation (Shi et al., 2016). To the best of our knowledge, the proposed HTM algorithm represents the first approach that exploits the hierarchical information in a Wikipedia article.

3 Hierarchical Trivia Miner

In this section, we describe the proposed HTM algorithm. Algorithm 1 shows the architecture of the HTM algorithm.

3.1 Top-down Processing

HTM receives a single Wikipedia article for the target entity and extracts trivia facts. It performs top-down processing using a surprise score (in Section 3.2) to extract trivia sentences, starting from a section to a sentence. Thus, HTM first ranks each section compared to the summary. The top-ranked section with the highest surprise score is the section that is considered to contain appropriate trivia facts, because the summary tends to contain well-known facts, and the dissimilarity from the summary can indicate the trivia-worthiness. Then, if the selected section includes subsections, HTM then ranks each subsection based on the summary. Otherwise, if the selected section does not contain subsections, the algorithm

Algorithm 1 Hierarchical Trivia Miner

```
1: function TOP-DOWN(ARTICLE, ENTITY)
2:     $B \leftarrow Summary$ in ARTICLE
3:     $S \leftarrow Contents$ in ARTICLE
4:     $N \leftarrow EntityName$ in ARTICLE
5:     $TriSec \leftarrow SURPRISE(B, S)$
6:     $TriSen \leftarrow None$
7:     while $TriSen \neq None$ do
8:         if $Subsec$ in $TriSec$ then
9:             $TriSubSec \leftarrow SURPRISE(B, TriSec)$
10:            if $Sub^2sec$ in $TriSubSec$ then
11:                $TriSub^2Sec \leftarrow SURPRISE(B, TriSubSec)$
12:                if $Sub^3sec$ in $TriSub^2Sec$ then
13:                    $TriSub^3Sec \leftarrow SURPRISE(B, TriSub^2Sec)$
14:                else
15:                    $TriPara \leftarrow SURPRISE(B, TriSub^2Sec)$
16:            else
17:                $TriPara \leftarrow SURPRISE(B, TriSubSec)$
18:        else
19:            $TriPara \leftarrow SURPRISE(B, TriSec)$
20:        $TriSen \leftarrow SURPRISE(B, TriPara)$
21:        if $ENTITY = True$ then
22:            $TriSen \leftarrow FILTERING(TriSen, N)$
       return $TriSen$
23: function SURPRISE(SUMMARY, CONTENTS)
24:     $B \leftarrow Summary$
25:     $S \leftarrow Contents$
26:     for $S_i$ in $S$ do
27:         Sim $\leftarrow SIMILARITY(B, S_i)$
28:         Surprise $\leftarrow \frac{1}{Sim}$
29:         TriviaScore.Surprise
30:     TriContent $\leftarrow \arg\max_i$[TriviaScore]
        return TriContent
31: function SIMILARITY(SUMMARY, CONTENT)
32:     $K \leftarrow 5$
33:     $T_1 \leftarrow TopTFIDF(Summary, K)$
34:     if $CONTENT = TriPara$ then
35:         $T_2 \leftarrow AllWordsInSentence$
36:     else
37:         $T_2 \leftarrow TopTFIDF(TEXT, K)$
38:     Similarity $\leftarrow \sigma(T_1, T_2)$
        return Similarity
39: function FILTERING(SENTENCE, ENTITY)
40:     if $Entity$ in $Sentence$ then
41:         Trivia $\leftarrow Sentence$
42:     else
43:         Trivia $\leftarrow None$
       return Trivia
```

proceeds downward to rank each paragraph in the selected section. Finally, the top-ranked sentence is extracted as a trivia fact from the selected paragraph.

To extract multiple trivia facts, HTM is performed recursively to extract additional trivia sentences after pruning the previously obtained ones.

3.2 Surprise Score

To perform the top-down processing from sections to sentences, we need a metric to obtain the top-ranked textual unit, a section, a subsection, a paragraph, and a sentence. Tsurel et al. (2017) showed that their surprise score can identify an unusual article among others based on the given article. Therefore, we also employ their surprise score to estimate how unusual the textual unit is compared to the summary.

We track an unusual textual unit from a section to a sentence using the surprise score. Here, we illustrate the process at the section level. We compute surprise scores between an article summary and each section in the given article. The surprise score of the section is defined as follows:

$$\mathrm{Surp}(B, S) = \frac{1}{\sigma(B, S)}, \tag{1}$$

where B denotes the summary and S denotes the section. σ is used to estimate the similarity between the summary and each section (in Section 3.3). Because the surprise score is the inverse of the similarity between the summary and the section, the section with the highest surprise score is the unusual and surprising section among sections, in that the summary consists of well-known and usual facts. The algorithm then proceeds recursively until the sentence level. The sections are ranked with the surprise score for the Confucius Wikipedia article in the following order: “Legacy”, “Life”, “Philosophy”, and “Name”. Here, the most surprising section, “Legacy”, contains subsections; thus, the HTM algorithm proceeds to compare surprise scores for the subsections to obtain a surprising subsection.

3.3 Similarity Metric

Term frequency-inverse document frequency (TF-IDF) reflects how important a word is in documents. Tsurel et al. (2017) used top-k TF-IDF words to compute article-article similarities and demonstrated that they can successfully compute the similarities for the surprise scores. Thus, we also employ high TF-IDF words to compute the similarity, $\sigma(B, S)$, in Equation (1), from a section to a sentence based on the summary. We first normalize a text in an article by removing stopwords (Tsurel et al., 2017) including the article name in calculating TF-IDF for words.

To capture semantic similarities, we employ word2vec embeddings, which are vector representations for words. We utilize pre-trained word embeddings based on the skip-gram model (Mikolov et al., 2013a) with the negative sampling method (Mikolov et al., 2013b). The word embeddings for approximately 100 billion word tokens mapped into 300-dimension vectors were learned using a part of the Google News corpus.

The obtained top-k TF-IDF words with their word embeddings are then used to compute the similarity for each section, subsection, paragraph, and sentence. Here, we weigh the obtained top-k words depending on their TF-IDF scores (Tsurel et al., 2017). For example, let T_1 and T_2 be a set of the top-k TF-IDF words for the summary and the section, respectively. We first search for the most similar term in T_2 for each high TF-IDF term in T_1. Similarly, we start from T_2 to search for the most similar term in T_1. Then, we sum these two similarities as follows:

$$\sigma(B, S) = \frac{1}{N} \sum_{i=1}^{k} w(i) \cdot (\max_{1 \leq j \leq k} c(t_1[i], t_2[j]) + \max_{1 \leq j \leq k} c(t_2[i], t_1[j])), \tag{2}$$

where B and S denote the article summary and the textual unit from a section to a paragraph, respectively. k is a hyperparameter for how many words we should use as the top-k TF-IDF words. t_1 and t_2 are the top-k TF-IDF words for the summary and the textual unit, respectively. $w(i)$ assigns linear weights to the top-k words, defined as ($k - i + 1$), and N normalizes the summed similarities, defined as $(k + 1)\,!$. $c(w_1, w_2)$ indicates the cosine similarity between two words, based on their word2vec embeddings.

Applying Equation (2) at the sentence level can result in unfair comparison if some sentences are very short because, in this case, a sufficient number of high TF-IDF words cannot be obtained from the sentence. Therefore, rather than using the top-k TF-IDF words for the sentence, we employ all words in it. Because we use all words, we remove $w(i)$ that assigns linear weights to the top-k words as follows:

$$\sigma(B, S) = \frac{1}{N} \sum_{i=1}^{k} \max_{1 \leq j \leq L} c(t_1[i], w_2[j]) + \frac{1}{Q} \sum_{i=1}^{L} \max_{1 \leq j \leq k} c(w_2[i], t_1[j]), \tag{3}$$

where B and S denote the article summary and each sentence in the surprising paragraph, respectively. L is the number of words in the sentence, and Q normalizes the summed similarities, defined as $(L+1)\,!$. w_2 is words in the sentence.

Calculating surprise scores based on high TF-IDF words might be inadequate if some named entities are included. Wikipedia articles tend to be collaboratively written about named entities, and so a large proportion of Wikipedia articles contain named entities (Nothman, 2008). Those named entities can be too abstract to calculate surprise scores due to their less informative embeddings, that may result in noises. To circumvent this issue, we consider another method for calculating similarities with top-k inverse document frequency (IDF) terms. Unlike TF-IDF scores that consider frequent words, IDF scores do not consider named entities that tend to appear frequently in each Wikipedia article. For example, Table 1 lists the top five TF-IDF and IDF words from the Stephen Hawking Wikipedia article. While words from the selected surprising section by TF-IDF scores include named entities, “Jane” and “Mason”, those from the selected surprising section by IDF scores does not contain named entities.

3.4 Trivia Candidate

The proposed HTM algorithm considers only the textual content in a Wikipedia article for a given entity. Therefore, sections “References” and “External links” were discarded in Figure 2. Furthermore, we omit the text written in other elements such as infoboxes, tables, images, categories, references, links, and itemized lists.

A trivia sentence mined by the proposed HTM algorithm may not always include a given entity token, that can reduce readability. Therefore, we adopt a filtering step here. If the mined trivia sentence does

Top 5 TF-IDF words		Top 5 IDF words	
Summary (T1)	Surprising section (T2)	Summary (T1)	Surprising section (T2)
cosmology	Jane	achieved	accused
achieved	Mason	breaking	acid
breaking	disabilities	cosmologist	action
cosmologist	drive	discusses	additional
discusses	family	English	afraid

Table 1: Top 5 TF-IDF and IDF words from the "Stephen Hawking" article. Words from the selected surprising section based on high TF-IDF words contain named entities ("Jane" is his first wife and "Mason" is his second wife.).

not contain an entity token, we remove the extracted sentence and recursively extract other trivia sentences until the entity token is found. It can improve the readability of extracted trivia facts with better understanding.

4 Experimental Settings

In this section, we describe our datasets for extracting trivia facts. We evaluate trivia facts mined from Wikipedia articles by using a crowdsourcing platform, Amazon Mechanical Turk.

4.1 Datasets and Parameters

For the test dataset, we prepared the most viewed Wikipedia articles from December 1, 2007 to January 1, 2019. This dataset included eight domains: countries, cities, people, music bands, sport teams, film and TV series, albums, and books. We randomly collected 100 articles in total, that were the similar size as the previous study (Tsurel et al., 2017).

To tune the hyperparameter k for high TF-IDF and IDF words in the proposed HTM algorithm, we used 10 random articles as the development dataset that were not present in the test dataset. Here, it was set to 5, based on human evaluation on the development dataset, from the range of 5 to 10.

4.2 Human Evaluation

We used Amazon Mechanical Turk to obtain the evaluation of extracted trivia facts. Each fact was presented to five human annotators, that are the same setting with the previous work (Tsurel et al., 2017; Prakash et al., 2015; Korn et al., 2019). The workers were given a target entity name, basic background, and an extracted trivia fact. For example, for Stephen Hawking, we provided the following background sentence: *"Stephen William Hawking CH CBE FRS FRSA (8 January 1942 – 14 March 2018) was an English theoretical physicist, cosmologist, and author who was director of research at the Centre for Theoretical Cosmology at the University of Cambridge at the time of his death"*. Note that the background is the first sentence in the Wikipedia article for the entity. Then, we provided the following extracted trivia fact: *"In August 2014, Hawking accepted the Ice Bucket Challenge to promote ALS/MND awareness and raise contributions for research."*. The workers were then asked with the following questions: "Is this statement a trivia fact to a user who is querying for the entity?" and "If the statement is a trivia fact, is it s good trivia fact?". The workers responded in the three-point rating scale ("Good Trivia fact", "Trivia Fact", "Not Trivia Fact").

Some sentences in Wikipedia articles are not understandable independently. Those sentences contain a pronoun that refers to a word in surrounding contexts. To deal with this issue, we manually changed such pronouns to proper words in the surrounding context to improve readability.

5 Experiments

In this section, we summarize the results of human evaluation for mined trivia facts. Then, we demonstrate that the proposed HTM algorithm can extract trivia facts regardless of target entity domains and does not incur high computational cost.

5.1 Comparison with Existing Methods

We compare the trivia facts mined by the proposed algorithm and the following existing methods:

- **Category (Baseline)** (Tsurel et al., 2017): Template-based trivia fact generation using the Wikipedia categories located at the bottom of articles.
- **Top-down using TF-IDF**: Trivia sentences are extracted using high TF-IDF words.
- **Top-down using TF-IDF + Filtering**: Trivia sentences are extracted using high TF-IDF words and then are filtered in cases without the target entity.
- **Top-down using IDF**: Trivia sentences are extracted using high IDF words.
- **Top-down using IDF + Filtering**: Trivia sentences are extracted using high IDF words and then are filtered in cases without the target entity.
- **Google algorithm** (Korn et al., 2019): Template-based trivia fact generation using Wikipedia's superlative tables. We manually searched each entity at the Google search engine to obtain the corresponding trivia facts.[1]

Table 2 summarizes the results of the human evaluation. The numbers shown in the table are how many facts are evaluated by the majority of human workers as the rating of "Good Trivia", "Trivia", and "Not Trivia", as well as the number of facts for "NoMaj", that indicates no majority among the workers. "Total" describes the number of available trivia facts from the test dataset. "Pronoun" describes the number of extracted trivia facts that are not understandable independently because of the existence of pronouns. "Cost" indicates the total elapsed time (in seconds) to mine 100 trivia facts in the test dataset.

For the comparison, we merge "Good Trivia" and "Trivia" together. The baseline method, **Category**, was proposed to target only at the people domain because articles from other domains generally do not include trivia categories. Thus, the variants of the proposed HTM algorithm, which consider Wikipedia's hierarchical structure, outperformed the baseline. Our best HTM algorithm using high IDF words recorded 42% better at mining trivia facts than the baseline method. In addition, our best HTM algorithm reduced the total elapsed time more than 110 times compared with the baseline. While HTM using high IDF words costed 118 seconds, the baseline required 12,956 seconds to extract trivia facts in the test dataset.

As we expected, using high IDF words to extract trivia facts showed better performance than using high TF-IDF words because high TF-IDF words can sometimes contain named entities, which can result in noises in top-down processing. While using high IDF words achieved the accuracy of 81%, using high TF-IDF words recorded the accuracy of only 71%.

Applying the filtering step does not guarantee improved performance of mining trivia facts. While the filtering step improved the performance of the proposed HTM algorithm in cases of using high TF-IDF words, it did not when using high IDF words. While 81% of facts from the HTM algorithm using high IDF words without the filtering step were considered as trivia by human evaluators, the HTM algorithm using high IDF words with the filtering step yielded only 76%.

Although trivia facts generated by the superlative tables from Google yielded high performance with the accuracy of 82.3%, the domain was quite limited. Trivia facts were available only for 17 out of 100 articles in the test dataset at the Google search engine. The coverage of the Google method seems rather small.

5.2 Analysis

We analyzed the evaluated trivia facts based on the individual domains of Wikipedia articles to demonstrate that the proposed HTM algorithm is domain-independent when extracting trivia facts. Figure 3 shows percentages of trivia facts for eight domains, extracted by HTM using high IDF words. They

[1]Google currently stops providing such trivia facts through knowledge panels, that are information boxes appearing as Google's search results.

Method	Good Trivia	Trivia	Not trivia	NoMaj	Total	Pronoun	Cost
Category	5	34	46	15	100	-	12,956
TF-IDF	25	46	13	16	100	8	117
TF-IDF + Filtering	25	50	9	16	100	2	469
IDF	**38**†	**43**†	8	11	100	9	118
IDF + Filtering	28	48	8	16	100	4	506
Google	12	2	1	2	17	-	-

Table 2: Human evaluation of trivia facts extracted by each algorithm. The best scores in all the proposed algorithms are shown in bold. † indicates that the improvement (in terms of the sum of "Good Trivia" and "Trivia facts") from the Baseline (Category) is statistically significant by using the paired bootstrap resampling method (Koehn, 2004) ($p < 0.001$).

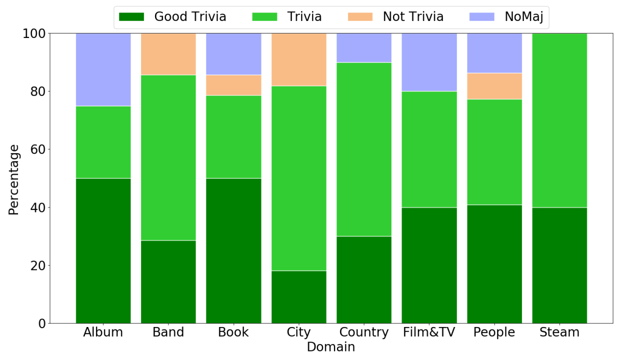


Figure 3: Percentages of trivia facts in eight domains from the test dataset.

consist of albums, music bands, books, cities, countries, film and TV series, people, and sport teams. As expected, the proposed HTM algorithm successfully mined trivia facts regardless of domains because the algorithm searches trivia facts through a single Wikipedia article of the given entity. The proposed HTM algorithm achieved nearly 80% accuracy (the sum of "Good Trivia" and "Trivia facts") of mining trivia facts for each of the eight domains.

Table 3 shows examples of good trivia facts extracted by the HTM algorithm using high IDF words. As shown, the extracted trivia fact for Billie Eilish is interesting. However, if we apply the filtering step, the example is no more trivia because it does not contain an entity token. Thus, recursively removing extracted trivia facts if they do not contain entity tokens might reduce the performance of extracting trivia facts.

6 Conclusion and Future work

In this paper, we showed that Wikipedia's hierarchical information can be applied for extracting trivia facts. We proposed *Hierarchical Trivia Miner*, which is domain-independent and does not incur high computational cost. Experimental results on eight domains demonstrated that the proposed HTM algorithm can extract "trivia facts" regardless of the target domains because it considers a single Wikipedia article for the target entity. In addition, the proposed algorithm employs top-down processing using hierarchical information, which can gradually narrow down its search space and reduce the total elapsed time to extract trivia facts.

The current filtering method discards mined trivia facts from the HTM algorithm if they contain no entity tokens. For the future work, we are planning to extend the filtering method by taking coreference resolution into account to address the problem of the current filtering method.

Domain	Entity	Good Trivia
Album	The Slim Shady LP	In the album's first week of release, The Slim Shady LP sold 283,000 copies, debuting at number two on the Billboard 200 chart behind TLC's FanMail and Britney Spears' debut ...Baby One More Time.
Band	Muse (band)	Most earlier Muse songs lyrically dealt with introspective themes, including relationships, social alienation, and difficulties they had encountered while trying to establish themselves in their hometown.
Book	The Lord of the Rings	The Lord of the Rings developed as a personal exploration by Tolkien of his interests in philology, religion (particularly Catholicism), fairy tales, Norse and general Germanic mythology, and also Celtic, Slavic, Persian, Greek, and Finnish mythology.
City	San Francisco	Geographically, Oakland Airport is approximately the same distance from downtown San Francisco as SFO, but due to its location across San Francisco Bay, it is greater driving distance from San Francisco.
Country	Indonesia	Indonesia has 8 UNESCO World Heritage Sites, including the Borobudur Temple Compounds and the Komodo National Park; and a further 19 in a tentative list that includes the Jakarta Old Town, Bunaken National Park, and Raja Ampat Islands.
Film & TV	The Dark Knight Rises	On July 20, 2012, during a midnight showing of The Dark Knight Rises at the Century 16 cinema in Aurora, Colorado, a gunman wearing a gas mask opened fire inside the theater, killing 12 people and injuring 58 others.
People	Billie Eilish	She was raised vegetarian and regularly advocates for veganism on social media.
Sports team	Inter Milan	The captain and coach of the first championship winning team was Virgilio Fossati, who was later killed in battle while serving in the Italian army during World War I.

Table 3: Examples of good trivia facts mined by the proposed HTM algorithm using high IDF words.

References

Janara Christensen, Stephen Soderland, Gagan Bansal, and Mausam. 2014. Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 902–912. Association for Computational Linguistics.

Nausheen Fatma, Manoj K. Chinnakotla, and Manish Shrivastava. 2017. The unusual suspects: Deep learning based mining of interesting entity trivia from knowledge graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1107–1113. AAAI Press.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Flip Korn, Xuezhi Wang, You Wu, and Cong Yu. 2019. Automatically generating interesting facts from wikipedia tables. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 349–361. ACM.

Shou-de Lin and Hans Chalupsky. 2003. Using unsupervised link discovery methods to find interesting facts and connections in a bibliography dataset. *SIGKDD Explor. Newsl.*, 5(2):173–178.

Matthew Merzbacher. 2002. Automatic generation of trivia questions. In Mohand-Saïd Hacid, Zbigniew W. Raś, Djamel A. Zighed, and Yves Kodratoff, editors, *Foundations of Intelligent Systems*, pages 123–130, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119. Curran Associates Inc.

Kazuya Niina and Kazutaka Shimada. 2018. Trivia score and ranking estimation using support vector regression and RankNet. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*. Association for Computational Linguistics.

Joel Nothman. 2008. Learning named entity recognition from wikipedia. Ph.D. thesis, The University of Sydney Australia 7.

Heather L. O'Brien and Elaine G. Toms. 2008. What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology*, 59(6):938–955.

Abhay Prakash, Manoj Kumar Chinnakotla, Dhaval Patel, and Puneet Garg. 2015. Did you know? - mining interesting trivia for entities from wikipedia. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 3164–3170.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics.

David Tsurel, Dan Pelleg, Ido Guy, and Dafna Shahaf. 2017. Fun facts: Automatic trivia fact extraction from wikipedia. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 345–354. ACM.

Ryen W. White and Susan T. Dumais. 2009. Characterizing and predicting search engine switching behavior. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 87–96. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.