Automatic Trivia Fact Extraction from Wikipedia

Qiancheng Liu
liu9204@tamu.edu

Aniket Bonde
bondeanikets@tamu.edu

Abstract

Searching is very important part of present internet era. Most of the search engines try to give you exact and only information you ask through the query. Providing additional interesting facts along with required information makes user search more exploratory and help in increasing the user engagement. Most random facts are not suitable for the trivia section. There is skill (and art) to curating good trivia. In this pro-ject, we formalize a notion of trivia-worthiness and implement an algorithm that automatically mines trivia facts from Wikipedia. We take advantage of Wikipedias category structure and rank an entitys categories by their trivia-quality. Our algorithm is capable of finding interesting facts, such as Obamas Grammy or Elvis stint as a tank gunner.

1 Introduction

Providing additional interesting facts along with required exact and only information you ask through the query makes user search more exploratory and help in increasing the user engagement. The latest search engines have started giving additional information to the user like www.yippy.com and others. We propose augmenting search results with trivia facts that are related to the searched entity. We believe trivia facts could contribute to the user experience around entity searches; even a small impact on this type of queries can translate into a signifi-cantly improved user experience Today trivia games are quite popular and played throughout world from pubs to mobile applications.

It has been found that more than 50 percent of web search queries are entity based. Hence amplifying the search result with such facts would help in increasing the user engagement and improve dwell time. Business case studies have shown that trivia helps improve revenue and user engagement. A man who tweets random facts has over 18 million followers and make about 500,000 dollars in a year.

In order to automate the process of finding good trivia, one needs to characterize the notion of what is trivia-worthy. In this work, we introduce and formalize two criteria that characterize good trivia: surprise and cohesiveness. Using our formulation, we propose an algorithm that automatically extracts trivia facts from Wikipedia arti-cles. We take advantage of the category structure of Wikipedia and rank an entitys categories by their trivia-quality.

2 Related work

There is relatively little work in Computer Science focusing on trivia. Some work includes using supervised learning to extract linguistic and entity-based features from a labeled dataset derived from the IMDb (Internet Movie Database) trivia section. Unlike our method, this algorithm does not utilize Wikipedias structure. In addition, its application is limited to domains where large free labeled databases such as IMDb exist. Despite being simpler, our algorithm finds better trivia facts. There is a large body of work devoted to the more general questions of surprise, interestingness and anomaly detection . For example, some develop a logicbased framework that translates structured news reports into formulas, identifying as interesting those that violate consistency or contradict axiomatic beliefs and expectations. Yet otherconsider the concept of interestingness as a users desire to know more about a topic. By observing webbrowsing logs of transitions between

Wikipedia articles they construct a probabilistic model that learns latent semantic features that are interesting to users.

Yet other conducts a literature survey of interestingness measures used in knowledge discovery, divided into objective statistical measures and subjective measures based on user beliefs or a specific domain. They define differential ratio rules to detect interesting patterns in spatio-temporal data. The technique uses ratios of features over time to detect change, similar to our definition of trivia-worthiness.

Miliarki et al. demonstrated a search module which explores entities related to a search query. It was proven to be an effective vehicle for drawing searchers to an exploratory activity. Interestingly, the highest engagement was registered when the mentioned entity was a person (as compared to location, or a movie). This serves as further motivation for our suggestion of augmenting entity search results with related trivia facts. An issue left open is how to predict the response in advance that is, whether the user is focused or exploratory.

3 Methodology

The goal is to automatically find trivia facts about entities. We first consider possible sources of such facts. Wikipedia is a natural choice for this purpose because of its wide coverage. However, Wikipedia articles are written in natural language; working with short textual units (e.g., sentences), one must deal with anaphora resolution, long-range references, and other context-related prob-lems. Thus, we focus on Wikipedias category structure. Categories are sets of articles with a shared topic, such as History of France, Philosophy of mind, or Biological concepts. An article can belong to multiple categories. For example, Barack Obamas categories include Presidents of the United States, Columbia Universi-ty alumni, and Grammy Award winners

Given an article, we want our algorithm to rank its categories, such that the top-ranked categories should be most suitable for the trivia section. Therefore, we need to formalize the notion of trivia-worthy. The Merriam-Webster dictionary defines trivia as Unimportant facts or details. Facts about people, events, etc., that are not well-known. One possible direction for detecting a trivia-worthy category would be to choose the category with the smallest number of articles. Presumably, a small category indicates a rare and unique property of an entity, and would be an interesting trivia fact. However, testing this path has shown it focuses on properties that were too narrow, in several senses: Most often, the smallest category focuses on a very specific identity aspect, usually obscure and uninteresting - Muhammad Ali is an alumni of Central High School in Louisville, Kentucky is not a good trivia fact - the specific high school has no importance to the reader and does not reflect on Alis character. In other cases, when the entity belonged to a well-known family, band or group, the smallest category captured a well-known aspect of the en-tity, for example Michael Jackson was a member of the The Jackson 5. Indeed, trivia facts are of-ten centered around uncommon knowledge. In other words, trivia facts are surprising. For ex-ample, everybody who knows Obama probably knows he belongs to the Presidents of the United States category, but many people would be surprised to learn that he won a Grammy. On the other hand, we want facts that are also interesting and not obscure.

4 Features

We have considered the following features for deciding if the fact is trivia-worthy:

1. Surprise

   Surprise measures how unusual it is for a given article to belong to a category. In other words, we would need to define a similarity metric between an article a and a category C. Since a category is a collection of articles, our main building block will be a similarity metric between articles.

   We defined a similarity metric between an article 'a' and a category 'C'. Since a category is a collection of articles, our main building block will be a similarity metric between articles. We denote article-article similarity by (a, a0). Next, we extend the similarity between articles to similarity between articles and categories. A category 'C' is a set of articles. We define the similarity of an article a to category 'C' as the average similarity between 'a' and the articles of C:

$$\sigma(a, C) = \frac{1}{|C| - 1} \sum_{a \neq a' \in C} \sigma(a, a') \quad (1)$$

An article is surprising w.r.t. a category if its average similarity to the other articles is low. Thus, we define surprise as the inverse of the average similarity:

$$surp(a, C) = \frac{1}{\sigma(a, C)} \tag{2}$$

2. Cohesiveness

   It is defined as the average similarity between the articles of the category. Cohesiveness describes the elusiveness of the facts.

   These features do not yet produce a good set of trivia facts. For that, we have to consider both surprise and cohesiveness:

   We define cohesiveness of category C as the average similarity between pairs of articles from 'C':

$$cohesive(C) = \frac{1}{\binom{|C|}{2}} \sum_{a \neq a'} \sigma(a, a') \tag{3}$$

We have just defined two properties surprise and cohesiveness. We now wish to combine them into a notion of trivia-worthiness

1. Trivia Score

   Figure 1 shows the categories of Barack Obama. The y-axis represents cohesiveness, and the x-axis represents surprise. Intuitively, a category is trivia-worthy if it is high on both surprise and cohesiveness scores. Therefore, we would like to define a score that is monotonic in both dimensions. We define the trivia-worthiness of a category 'C' w.r.t. article 'a' as the product of cohesiveness and surprise.

$$trivia(a, C) = cohesive(C) * surprise(a, C) \tag{4}$$

The Figure 1 shows cohesiveness vs surprise for various categories belonging to Barack Obama. As we can see again Grammy award winner category has both high cohesiveness and surprise as compared not so interesting category of Washington DC Democrats.

There are many other ways to combine the two scores. However, multiplication has a natural interpretation. Note that surprise is defined as the inverse of the average similarity of a to the category. Thus, the multiplicative formula measures how similar the article is to the category, compared to the average similarity of articles from the same category. In other words, we measure whether the article is more similar or less similar to the category than was expected.

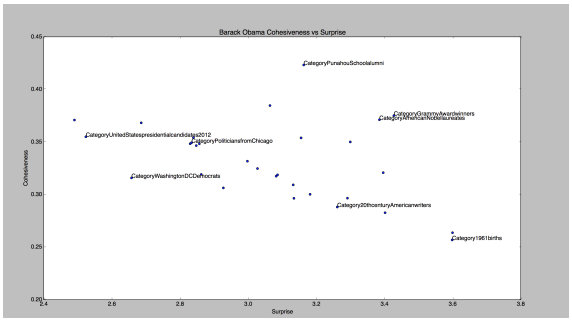


Figure 1. cohesiveness vs surprise plot for Barack Obama.

A value of trivia around one means, by definition, that the average distance between a and C is similar to the cohesiveness of C, which indicates that the article is typical for that category: it is similar to other articles in the category just as much as the average article. A value of trivia much lower than one indicates that the article is more similar to other articles than the average, and could be an exemplar. It is a prominent member of the category, and would not be good trivia. Now, a value of trivia higher than one means that the article in question is not so similar to the category. In some sense, it is an outsider, which might make good trivia.

5 Overall Architecture of the Project

1. WikiParser

   The module uses pywikibot to get the data from the wikipedia entities and categories related to it. We have used wiki2plain interface for converting the wiki data into raw text format.

2. Wiki Trivia Metric Calculator

   This module calculates the metrics such as top K TF-IDF and entity-entity similarity values for the wikipedia entities.

3. Algorithm

   This is the brain module of the project that gets data using wiki parser or from cached data and then computes the ranked trivia worthy facts about the desired entity.Figure 2 is the overall architecture for our back-end.

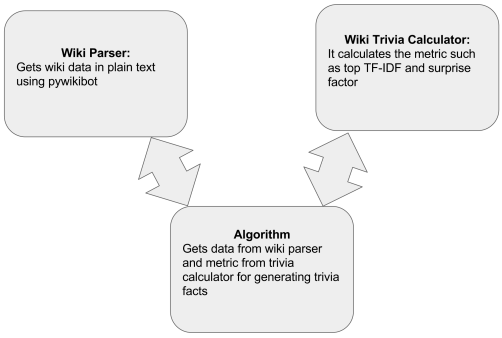


Figure 2. Overall architecture for our backend.

We have used the pywikibot and wikipedia libraries for scrapping the data from the wikipedia.

Once we get the input entity from the user for which we have to calculate the Trivia facts. For the first case we will check whether the entity exists in the wikipedia or not. Sometimes it might happen that the user typed wrong spellings or half name, so we use the wikipedia search api for getting the best and correct entity name that is an article in the wikipedia here.

Once we got the article entity here, we will proceed with the fetching the article text using the pywikibot api. We also get the categories that entity belong to.

The next step involves the preprocessing of the retrieved data from the wikipedia. First of all we need to remove the html tags and get the plain text from it. We have used Wiki2Plain[7, 21] code for removing the html tags, images, punctuations and also unwiki the text, as it contains the text in some format like sections and subsection. The second step of preprocessing is lowercase all the text so that we have uniform cases. The third step is to remove the stop words. we have used the nltk library to remove the stop words. And the final preprocessing step is to do stemming using Porter Stemmer of the words and get the tokens for each article.

To enhance the computation of our algorithm we have used the cache at different levels. We have cached the stemming results. Moreover to reduce the api calls to fetch the data from wikipedia every-time we need the said article, we cache the top k tokens of the said entity/article in a file using the name of entity, so that we can directly fetch the token if it already has been processed. This helped us in saving alot of computation and api calls to wikipedia which is throttled. Also we are storing the final ranked trivia facts, for supporting the offline ranked retrieval of trivia facts.

6 Detailed Algorithm

The detailed Trivia Extraction Algorithm is given below. It contains three main function algorithms called from "TriviaExtract" function:

Algorithm 1 Trivia Extract

function TRIVIAEXTRACT($inputArticle$)
  for every category C of $inputArticle$ **do**
    $surp = Surprise(inputArticle, C)$
    $cohes = Cohesiveness(C)$
    $C.trivia = cohes \ . \ surp$
return category C with maximum trivia score

Algorithm 2 Surprise

function SURPRISE($inputArticle, C$)
  $sum, count = 0$
  for $every\ article\ a \neq inputArticle\ in\ category\ C$ **do**
    $sim = ArticleSim(inputArticle, a)$
    $sum = sum + sim$
    $count = count + 1$
  $similarityToCategory = sum/count$
  $surprise = 1/similarityToCategory$
return surprise

Algorithm 3 Cohesiveness

function COHESIVENESS($category$)
  $sum, count = 0$
  for $every\ article\ pair\ a1 \neq a2\ in\ category\ C$ **do**
    $sim = ArticleSim(a1, a2)$
    $sum = sum + sim$
    $count = count + 1$
  $cohesiveness = sum/count$
return cohesiveness

We do not use all words in an article. Instead, we compute TF-IDF scores for all words in the documents. TF-IDF measures how important a word is in a document, given a corpus.

For our corpus, we used a sample of 10, 000 articles from the English Wikipedia. We used standard text normalization techniques such as stemming, stopword removal and case folding. We removed terms appearing in less than 10 documents.

Algorithm 4 Article Similarity

1: **function** ARTICLESIM($article1, article2$)
2: $K = 10$
3: $T1 = TopTFIDF(article1, K)$
4: $T2 = TopTFIDF(article2, K)$
5: $sim = (article1, article2)$
return sim

7 Evaluation

We evaluate our algorithm empirically. We have compared the following algorithms:

1. Wikipedia Trivia Miner (WTM):

   A ranking algorithm over Wikipedia sentences, which learns the notion of interestingness using domain-independent linguistic and entity based features. The supervised ranking model is trained on existing user-generated trivia data available on the Web.

2. Top Trivia:

   The highest ranking category in our algorithm ranking.

3. Middle-ranked Trivia:

   Using middle-of-the-pack ranked categories, as ranked by our algorithm.

4. Bottom Trivia:

   Using the lowest-ranked categories by our algorithm. We collected article and category data for our experiments via the Wikipedia web API using the Pywikibot framework and an adapted version of the Wiki2Plain interfac The best way to evaulate our Trivia Miner is to run user studies and find if they find our system to be interesting. Due to lack of time we were not able to run user studies.

To know if our algorithm is generating correct results , we compared trivia facts for entities like Barack Obama and Bill Gates, to trivia facts generated by above systems and found that there were similar trivia generated by both systems (just comparing top 5). However we didnt get the 100 percent overlap, that may be due random of sampling k entities for category or difference in implementation. However we found facts like Grammy Award for Obama and Pudma Bhushan for Bill gates to interesting and less know, hence it qualifies for Trivia. We analyzed trivia facts for Donald Trump like WWE Hall of Fame and for Lionel Messi to be part of UNICEF and People convicted of fraud to be quite amusing.

8 Conclusion and Future Work

Majority of the search engines enlist the results which most of the people know or plethora of the web-pages reflect thus always passing the common and popular results. Moreover, the search engines results try to fulfill the information need of the user. But it might be fun and learning experience to supplement this information with trivia facts, which might surprise and entertain you. In the work presented above we provided a metric to calculate the trivia-worthiness of the facts obtained from the wikipedia. Since the evaluation part of the work couldn't be performed given the time-lines and the fact that a trivia can be subjective- meaning depending in the person. This work has many benefits,as Tsurel puts it "while seemingly lighthearted, can lead to higher engagement of users searching for named entities. If successful, even a small impact on this type of queries can translate into a substantial improvement in user experience, and possibly transfer to other domains of human activity, like education.".

Furthermore, to engage the user in the trivia facts and to see how our algorithm works, we developed a game based on the trivia obtained using our algorithm. We take the top five trivia facts for each entity our system has seen till now and then develop five True or False questions. Since we know the ground truth about the entity and its corresponding trivia. We form the question based on the ground truth, we involve the False answer based question by shuffling the entity and corresponding category so that now an entity can either exist in the category or not. As we know the correct answer whether that person or entity exists in the category we can easily evaluate the answer and give a score to the user. This helps to enhance the trivia knowledge related to the random person or entity.

The prevailing view in the information-retrieval community sees the search engine as a passive librarian, looking up the facts. However, many users today expect the search engine to provide not just information, but also entertainment. We believe that with the advent of new search interfaces, it is time to re-examine the idea of adding serendipity to search. Building on the popularity

of entities (and in particular person entities) in current search, we propose an algorithm to identify facts about people as trivia-worthy. Specifically, we examine group membership in Wikipedia categories and rank them according to two dimensions: surprise and cohesiveness.

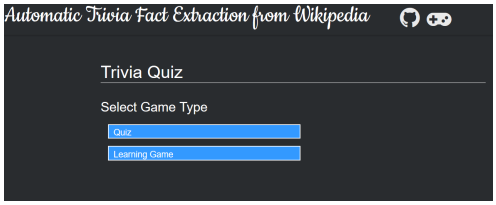
Future work can be in two directions, one taking the data from some other source. As we know that the wikipedia is crowd sourced website and the chances of a category being a fact about an entity/person is debatable. Other direction could be coming up an new metric to calculate the trivia-worthiness of the fact.

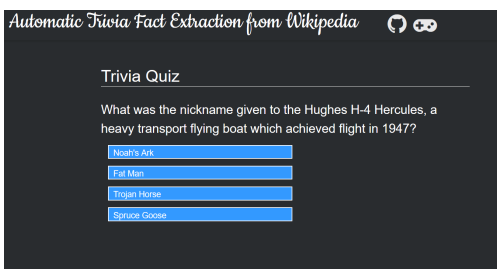
This application, while seemingly lighthearted, can lead to higher engagement of users searching for named entities. If successful, even a small impact on this type of queries can translate into a substantial improvement in user experience, and possibly transfer to other domains of human activity, like education.

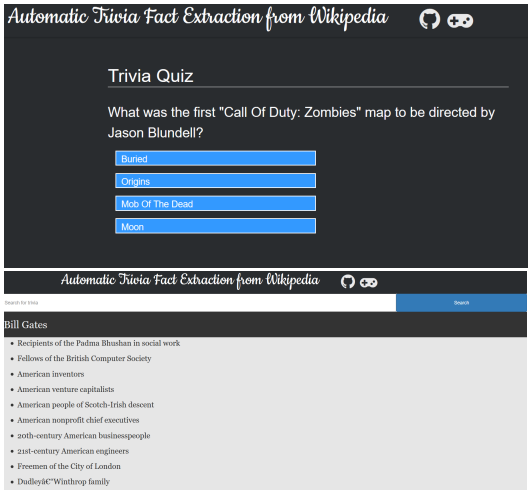
9 App!

We have created an app from our interesting results. The project's github link: https://github.com/qiancheng6/Automatic-Trivia-Fact-Extraction-from-Wikipedia

Use the README.md there to run the app. Following are some of the snapshots from our application.







References

Mika, Peter. "Entity search on the web." Proceedings of the 22nd International Conference on World Wide Web. ACM, 2013.

Yin, Xiaoxin, and Sarthak Shah. "Building taxonomy of web search intents for name entity queries." Proceedings of the 19th international conference on World wide web. ACM, 2010.

Sergey Chernov, Tereza Iofciu, Wolfgang Nejdl, and Xuan Zhou. Extracting semantics relationships between Wikipedia categories. SemWiki, 206, 2006.

Abhay Prakash, Manoj K. Chinnakotla, Dhaval Patel, and Puneet Garg. Did you know?: Mining interesting trivia for entities from wikipedia. In Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, pages 3164–3170. AAAI Press, 2015.

Matthew Merzbacher. Automatic generation of trivia questions. In International Symposium on Methodologies for Intelligent Systems, pages 123-130.Springer, 2002.

Tsurel, David, et al. "Fun Facts: Automatic Trivia Fact Extraction from Wikipedia." WSDM 2017.

Marco Baroni, Georgiana Dinu, and Germ an Kruszewski. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of Association for Computational Linguistics (ACL), volume 1, 2014.

Tom Kenter and Maarten de Rijke. Short text similarity with word embeddings. In Proceedings of the 4th ACM International on Conference on 353 Information and Knowledge Management, CIKM 15, pages 14111420, New York, NY, USA, 2015. ACM.

Wiki2Plain:http://stackoverflow.com/questions/4460921/extract-the-first-paragraph-from-a-wikipedia-article-python

MediaWiki. Manual:Pywikibot Mediawiki, The Free Wiki Engine, https://www.mediawiki.org/w/index.php?title=Manual:Pywikibotoldid=2176177, [Online; accessed 17-July-2016].

This 25-year-old makes $500,000 a year tweeting random facts, http://www.cnbc.com/2016/07/16/25-year-old-kris-sanchez-makes-500000-a-year-from-uberfacts.html. [Online; accessed 17-July-2016]

Ken Jennings. Brainiac: adventures in the curious, competitive, compulsive world of trivia buffs. Villard Books, 2007.