Tabouid: a Wikipedia-based word guessing game

Timothée Bernard
National Institute of Advanced Industrial Science and Technology (AIST), Japan
timothee.bernard@ens-lyon.org

Abstract

We present Tabouid, a word-guessing game automatically generated from Wikipedia. Tabouid contains 10,000 (virtual) cards in English, and as many in French, covering not only words and linguistic expressions but also a variety of topics including artists, historical events or scientific concepts. Each card corresponds to a Wikipedia article, and conversely, any article could be turned into a card. A range of relatively simple NLP and machine-learning techniques are effectively integrated into a two-stage process. First, a large subset of Wikipedia articles are scored — this score estimates the difficulty, or alternatively, the playability of the page. Then, the best articles are turned into cards by selecting, for each of them, a list of banned words based on its content. We believe that the game we present is more than mere entertainment and that, furthermore, this paper has pedagogical potential.[1]

1 Introduction

Thanks to its considerable size — a total of more than 50 million articles in 300 different languages today — and its availability online, Wikipedia has found many uses other than those of the traditional encyclopaedia. It is indeed frequently used for research in AI and natural language processing (NLP). For example, various large-scale machine-readable knowledge bases have been generated from the online encyclopedia, including YAGO (Suchanek et al., 2007), YAGO2 (Hoffart et al., 2013) or DBpedia (Bizer et al., 2009), in addition to reading comprehension datasets such as SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017). The plain text from Wikipedia articles has also been used directly as the only source of knowledge for Question-Answering systems such as the one developed by Chen et al. (2017).

This article presents a system which uses Wikipedia to generate the content of an application that is inspired by Taboo, a word-guessing board game originally published by Parker Brothers in 1989. To play our version of the game, called "Tabouid", all the group (of at least two) players require is a single electronic device (typically a smartphone). The game is divided in turns. During her turn (the length of which is defined by a countdown), the player looks at the *card* displayed on the screen. A card is composed of a *title* and a list of additional *banned words* (the words included in the title are considered banned words). See Figures 1 and 2 for two screenshots of the application (the circle around the title of the card acts as a countdown). The player sets out to make the other players guess the title on the card (in its exact wording). To do so, she has to describe the concept to the other players but without using any of the banned words, nor any words constructed with the same stems (translations of the words into other languages are not allowed either). Once a title has been guessed, the player continues on to the next card. The player has to skip the current card as soon as she mentions a banned word.

The originality of Tabouid lies in the fact that its content has been automatically generated from Wikipedia using a range of NLP and machine learning techniques. This automated process means that Tabouid can benefit from a wealth of 10,000 cards in English, and as many in French, covering not only words and linguistic expressions but also a variety of topics including artists, historical events or scientific concepts. In addition, all cards in Tabouid are associated with a difficulty score. This allows the difficulty level of the game to be set in a straightforward way. With such an adaptable difficulty, the game can accommodate various groups of players,

[1]The work presented in this paper is a personal project and is not directly related to my research at AIST.

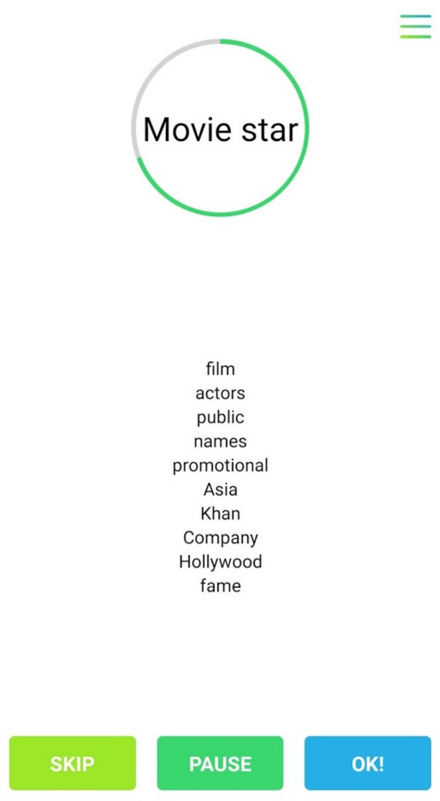

Figure 1: Screenshot of the application with the card `Movie Star`.

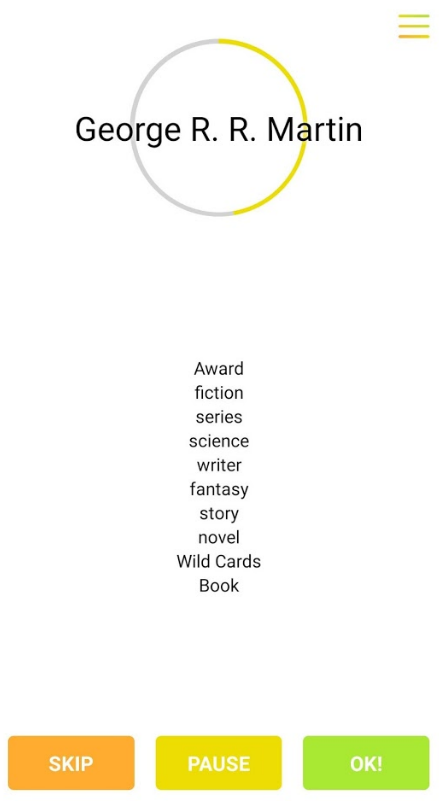

Figure 2: Screenshot of the application with the card `George R. R. Martin`.

which could include individuals such as children or foreigners, whose level of proficiency or knowledge of the culture associated with the target language may vary. In addition, we believe that the work presented here can have some pedagogical value by defining an implementation project for students enrolled in NLP or computational linguistic programs.

In this article, we describe how the cards were generated. Each card corresponds to a Wikipedia page. The title of the card is the title of the page, and the additional banned words (or expressions) are extracted from the text (and links) of the page. The automatic process is divided into two parts. First, we compute a difficulty score for each page of (a large subset of) Wikipedia as described in Section 2. Then, for the best pages (i.e., the easiest ones), we select a list of banned words as described in Section 3.

You can currently play Tabouid in English and French. The game is available as an Android[2] and iOS[3] application that is entirely free and does not contain any advertisement. The cards have been pre-generated and packaged with the application; an internet connection is not required during the game.

2 Page scoring

The computation of difficulty scores is the most complex, and probably also the most crucial, part of the process. Indeed, almost all of the pages in Wikipedia would make for very poor cards, and so represent little to no interest to the game. They broadly fall into two categories. The first category is that of pages referring to entities lying very far from what we can consider *common knowledge*. For example (taken randomly from Wikipedia), `Smrekovec Lodge`, described as “a mountain hostel on the southern slope of the Smrekovec Mountains” (in Slovenia). The other category comprises pages that do refer to relatively familiar concepts, but that are way too specific and technical to be actually guessed in the context of the game. For example, `Taekwondo at the 2016 Summer Olympics – Men's +80 kg`. Wikipedia

[2] https://play.google.com/store/apps/details?id=com.tot.tabouid

[3] https://apps.apple.com/us/app/tabouid/id1477994156

contains a very large number of pages for sport events of all kinds and in many variations — while some hardcore sport fans might actually enjoy playing such cards, we will assume than most players would quickly get bored after having a good laugh (at best).

By contrast, here are several examples of pages that will make for good cards: `Saturday`, `Feminism`, `Christopher Columbus` and `Bangladesh`. In the middle, pages such as `List of scientists who disagree with the scientific consensus on global warming` and `History of mathematical notation` can be found, which are definitely challenging but still playable and fun for individuals used to the game. We therefore want to compute difficulty scores that reflect this natural gradient.

To do so, we use a neural network that takes as input a vector representation of a page and computes a real-valued score between 0 and 1. The lower the score, the more difficult the page.[4] We first start by describing how the vector representation of each page is computed before turning to the neural network itself before describing the annotation collection.

2.1 Page representation

Each page vector can be divided in two subvectors: a vector of *categories and title features*, containing information about both the categories the page belong to and its title[5], and a vector of *various features*.

There are 10 various features: (1) the number of visits to the page between January 1st 2015 and December 31th 2018, (2) the variance of the monthly distribution of the number of visits between these two dates divided by its squared mean (this is a measure of dispersion of the distribution), (3) the size of the page, (4) the date of creation of the page, (5) the date of last modification of the page, (6) the number of modifications of the page since its creation, (7) the number of translations of the page in other versions of Wikipedia, (8) whether the title contains the name of a month, (9) whether the title is a date (matching some regular expression), and (10) whether the title contains foreign characters. Except for the three last of these, which are binary, each feature is formatted as a real value which is then linearly transformed so that the corresponding distribution over all pages has mean 0 and standard deviation 1.

The *categories and title* vectors are computed as follows. We first build the *title word assignment matrix* W containing one line for each page and where rows represent words, such that $W_{i,j} = 1$ if the title of page i contains word j and 0 otherwise.[6] We also build the *category assignment matrix* C in a similar way, such that $C_{i,j} = 1$ if page i belongs to a category containing word j, and 0 otherwise. Then, we concatenate these two matrices and apply a dimensionality reduction algorithm. More specifically, we use the `TruncatedSVD` algorithm implemented in the Scikit-learn library (Pedregosa et al., 2011) to produce a vector of size 50 for each page.[7]

2.2 Neural network

We use a very simple architecture for the neural network. The reason is that, as explained below, we have very little annotated data to train the system on. Therefore, to prevent overfitting, we define a model with a small number of parameters and such that some of these parameters are (to a certain extent) *interpretable*. One of the advantages of interpretable parameters is that we can manually set them to sensible values at the beginning of the training process. By doing so, we want to encourage the network to leverage actual correlations of the underlying distribution rather than mere artefacts of the training set.

Let u be a page vector (of dimension 60). It is first sent through a square affine layer, $u' = A \cdot u + B$, the result of which is then used to perform an element-wise multiplication, $v = u * u'$. Finally, this vector is sent through a sigmoid layer of height 1 to produce the difficulty score, $s = \sigma(a \cdot v + b)$.

This model contains only 3,844 parameters and

[4] Alternatively, the score can be interpreted as a *playability* score: the higher the score, the higher the playability of the page.

[5] In Wikipedia, each page belong to zero, one or more *categories* such as `20th-century women scientists`, `Naturalized citizens of France` or `People from Warsaw`, to name a few of the 58 categories the `Marie Curie` page belongs to. These categories define a hierarchy that is at the heart of knowledge-bases such as (Suchanek et al., 2007) and YAGO2 (Hoffart et al., 2013), in addition to taxonomies, such as WikiTaxonomy (Ponzetto and Strube, 2011) and WikiNet (Nastase et al., 2010).

[6] We ignore stop words and words appearing in less than 0.1% of the page titles. We use a set of one million randomly selected pages.

[7] This algorithm is based on singular value decomposition (SVD; Halko et al., 2011), which is also at the heart of latent semantic analysis (LSA; Dumais et al., 1988).

allows direct multiplicative interaction between each pair of coefficients of the input vector. In addition, if we expand the score as

$$s = \sigma(\sum_i (a_i u_i B_i + a_i u_i \sum_j A_{i,j} u_j) + b) \quad (1)$$

we can notice the linear $a_i u_i B_i$ terms. We recall that, for $0 \leq i \leq 9$, we know exactly what u_i means.[8] For all of them, we can intuitively guess whether they have a positive or negative impact on the difficulty or playability of the corresponding card, which allows us to initialise the product $a_i B_i$ in a sensible way. For example, we assume that, all other things being equal, the higher the popularity of the page, the lower the difficulty of the card. Similarly, a high dispersion in the visitor distribution might indicate a temporary fame of the page, which would have possibly made an interesting card for a short period after the peak of the distribution, but likely to become obsolete after that. So, we initialise $a_0 B_0$ with a positive value and $a_1 B_1$ with a negative one. More precisely, we initialise $a_i = 1$ for all i and then set $b_0 = 1$, $b_1 = -0.5$, $b_2 = 1$, $b_3 = -1$, $b_4 = 0.5$, $b_5 = 0.5$, $b_6 = 1$, $b_7 = -0.5$, $b_8 = -0.5$, $b_9 = -1$. Note that these weights will be trained with all other parameters, potentially (in)validating our intuitions.

Given a training set $D = (x_i, y_i)_i$ where x_i is a Wikipedia article and y_i is its annotated score, we train the model to minimise the cross-entropy over D, $L = -\sum_i s(x_i)\log(y_i) + (1 - s(x_i))\log(1 - s(x_i))$. The training is done by stochastic gradient descent with momentum, using L2 regularisation.

2.3 Annotation and results

To train the model described above, we collect annotations for a very small subset of Wikipedia. Pages are annotated with a real-valued score between 0 (hard/unplayable) and 1 (easy). Because, as explained above, Wikipedia is strongly imbalanced towards unplayable pages, manually annotating random pages chosen from a uniform distribution would be a very inefficient process. Instead, we implement an *active learning* process.

In short, we start by scoring the 100 most visited pages, as they contain a high proportion of easy pages. This forms our initial dataset. We train the model on this dataset and use it to score some unlabelled pages among which we select some instances to be scored by a human annotator. We add these 10 instances to the dataset and reiterate.

The query strategy that we adopt is the following: if the average score in the dataset is below 0.5, we select the 10 pages with the highest predicted score, otherwise we select the 10 lowest ones. The rational of this choice was to keep the dataset balanced. In retrospect, more principled techniques such as *expected model change* or *uncertainty sampling* (Settles, 2009; Fu et al., 2013) might have been tried, but this basic strategy yielded satisfying results, so we stuck to it.

After having annotated around 2300 instances in this way, the 500 pages with highest predicted scores were sent to human annotators. Once this process was completed, we trained the model on 70% of the scored pages, dividing the remaining 30% for early stopping and evaluation. One possible way to quantify the performance of this model is to discretise the score space ($[0, 1]$) into two categories ($s \leq 0.5$ and $s > 0.5$) and to compute the accuracy as in a binary classification task. On 10 trained models with the interpretable weights initialised as explained above, the best accuracy is 85.0%, the mean accuracy is 83.5% and the standard deviation is 1.1%. On 10 trained models without the initialisation procedure, the best accuracy is 84.4%, the mean accuracy is 82.9% and the standard deviation is 1.3%. This tends to show that our initialisation procedure is justified and makes the training more effective and reliable.

By manually inspecting the weights of a randomly chosen trained network, we can compute the $a_i B_i$ products for the 10 various features. We will not comment on most values, which are unsurprising but only mention that contrary to our expectations, the product for the size feature is very low ($a_2 B_2 = 0.03$) and the one for the number of translations too ($a_6 B_6 = 0.02$). This does not mean, however, that these features do not have a positive effect on the score of the page, as they are also involved in the multiplicative terms (see 1).[9]

The 10,000 pages with the highest scores (this includes pages annotated by humans) — from `Donald Trump` (1.0) to `Landscape painting` (0.59) — have been included as cards in the game. The selection of the banned words for

[8]The interpretability of the other coefficients is dependent on the result of the dimensionality reduction, which we will not discuss here.

[9]See for instance the work of Lipton (2018) about the difficulty of interpreting even simple models.

each page is the subject of the next section. The players can set a difficulty setting that determines which cards are shown during the game: if the difficulty is set as $d \in [1, 10000]$, only the d easiest cards are used (in random order). This setting, however, has no effect on each card's list of banned words.

3 Banned words selection

A simplified version of our banned words selection algorithm is as follows. For a given Wikipedia article, its text is first tokenised. Tokens are then stemmed and all stop words are removed. Finally, we select the 10 most frequent stems in the page and use their most frequent tokens (one for each) as banned words. Tokenisation and stemming are performed with the NLTK library (Bird et al., 2009).

There are two main differences between this simplification and the actual algorithm we use. The first is that we add rules to NTLK's stemmer in order to map strongly related words that are stemmed differently to the same class. For example, we send words stemmed as *lawyer* to the *law* class. We send the words stemmed as *pole* or *poland* (but starting with a capital letter) to the *polish* class. For all stems ending in *pean*, we remove the final *ean* (e.g., *European* is sent to *europ*). Around 25 rules have been manually defined during the development of the English version of the game.

Second, we do not ban only single words, but also longer expressions. We detect links pointing to other Wikipedia articles and consider their titles as potential banned expressions. Also, because it appears that entities having a Wikipedia page of their own are usually very informative even when they have comparatively few mentions in the text, we count them with a factor 1.5. Given the name of a Wikipedia article linked in the current page, each of its occurrences counts not only as 1 occurrence for each of the tokens it is composed of but also as 1.5 occurrences of the full name. For example, each occurrence of *Serge Gainsbourg* will count as 1 occurrence of *Serge*, 1 occurrence of *Gainsbourg* and 1.5 occurrences of *Serge Gainsbourg*. An occurrence of a single word entity name, such as *Madonna*, simply counts as 1.5 occurrences of *Madonna*. During the final step of the algorithm, when selecting the banned words or expressions, were two words composing the name of an entity to be selected, we only select this name instead.

This algorithm is not perfect; some important words might be missed because of the way we analyse the page or simply because they do not even appear in the text. In addition, even with our hand-crafted rules supplementing the stemming algorithm, different forms of the same word (or words that, although different, are so strongly related that, according to the rules of the game, banning one is equivalent to banning the other) might be selected as distinct banned words. It is to compensate for such limitations that we build lists of 10 banned words or expressions. Such a relatively high number tends to favour the introduction of false positives, but these are not a major problem. They are only an annoyance in that they unnecessarily slow the player, who has to read them. While we have not performed any quantitative evaluation of this algorithm, it has been, however, extensively tested during various parties, family gatherings, commuting trips and scientific conferences (among others). In addition to the two cards shown in Figures 1 and 2, here are a few other cards present in the game. `World War I` (score 1) with banned words *France*, *Russia*, *British Army*, *forces*, *Allied Powers*, *Battle*, *Britain*, *Ottoman*, *Germany*, and *United States*, `Rihanna` (score 1) with banned words *album*, *music*, *released*, *single*, *Billboard*, *Girl*, *song*, *record*, *Awards*, and *featured*, and `Artificial intelligence` (score 0.98) with banned words *AI*, *human*, *machine*, *research*, *learning*, *computer*, *problems*, *systems*, *networks*, and *algorithms*.

4 Conclusion

In this paper, we have shown how a range of relatively simple NLP and machine-learning techniques can be integrated effectively to automatically generate the content of Tabouid, a word-guessing game freely available on Android and iOS devices. Although easy to understand and implement, these techniques can be developed and improved on in many ways. They also naturally lead to a wide range of practical and theoretical questions relevant to NLP (e.g., data collection and annotation, and model interpretability). In this respect, this work could inspire implementation projects in NLP or computational linguistic programs. Concerning the game itself, we believe that Tabouid is more than just a fun game and can develop and help reinforce general knowledge for players of all backgrounds. It also appears to be an engaging way to practice speaking for language learners.

In addition to improving the banned words selection process, future work on Tabouid includes generating specific lists of cards based on school programs to use the game as an educational tool, using the category system of Wikipedia to let users select more or less specific categories to play with, and adapting the algorithms to leverage the wide variety of languages Wikipedia is available in beyond English and French. Currently, the content of Tabouid aims to reflect the diversity of Wikipedia's encyclopaedic knowledge." As a consequence, some cards include words related to topics that might be deemed inappropriate for children. As suggested by an anonymous reviewer, another possible addition to the game could then be to predict the age appropriateness of a given topic, allowing for cards to be filtered on the basis of an age setting.

Acknowledgements

Many thanks to Élise Dessaux for her work on the graphic design of the application!

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 281–285, Washington, D.C., USA. Association for Computing Machinery.

Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283.

N. Halko, P. G. Martinsson, and J. A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Zachary C. Lipton. 2018. The Mythos of Model Interpretability. *ACM Queue*, 16(3):1–27.

Vivi Nastase, Michael Strube, Benjamin Boerschinger, Caecilia Zirn, and Anas Elghafari. 2010. WikiNet: A Very Large Scale Multi-Lingual Concept Network. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Simone Paolo Ponzetto and Michael Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175(9):1737–1756.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM. Event-place: Banff, Alberta, Canada.