Assessing Word Difficulty for Quiz-Like Game

Jakub Jagoda and Tomasz Boiński(✉)

Department of Computer Architecture, Faculty of Electronics, Telecommunication and Informatics, Gdańsk University of Technology, Gdańsk, Poland
tobo@eti.pg.gda.pl

Abstract. Mappings verification is a laborious task. Our research aims at providing a framework for manual verification of mappings using crowdsourcing approach. For this purpose we plan on implementing a quiz like game. For this purpose the mappings have to be evaluated in terms of difficulty to better present texts in respect of game levels. In this paper we present an algorithm for assessing word difficulty. Three approaches are presented and experimental results are shown. Plans for future works are also provided.

1 Introduction

During our research, mainly during the Colabmap[1] project we created a set of mappings between English Wikipedia articles and WordNet synsets [1–4]. Each mapping consists of a WordNet synset, definition of the synset and the title of Wikipedia article with special characters encoded using RFC 3986. Such mappings, when proved to be correct, will allow formalization of Wikipedia structure. The obtained set of mappings contained algorithmically created 54475 connections. We aim at creating 100% correct mappings corpora so the set required manual verification.

In 2006 Luis von Ahn proposed usage of computer games as something more than pure entertainment and thus creating the idea of so called GWAP (Game With A Purpose) [5]. GWAPs are typical games that provide standard entertainment value that users expect but are designed in a way that allows generation of added value by solving a problem requiring intellectual activity. It is worth noticing that GWAPs does not allow financial gratification for the work. The will to continue playing should be treated as the only way of gratifying users [6]. Tempted by the results obtained during the Samsung's survey we decided on implementing a GWAP for validation of those connections [7] following the Human Based Computation model [8].

The originally obtained mappings were extended with three additional "next best" mappings with the idea of presenting the user a question (definition of a synset) with 4 possible answers (Wikipedia article titles). At the beginning the 3 other answers were randomly selected from the set of Wikipedia's pages but

[1] http://kask.eti.pg.gda.pl/colabmap.

J. Szymański and Y. Velegrakis (Eds.): IKC 2017, LNCS 10546, pp. 70–79, 2018.
https://doi.org/10.1007/978-3-319-74497-1_7

such approach quickly proved to be incorrect as the "next best" mappings were not related at all to the question. Instead we used Wikipedia search functionality to select alternative answers (according to Wikipedia). For the purpose of verification we implemented a 2D platform game called TGame[2] ("Tagger Game") as a 2D platform game following the output-agreement model [9].

In the next step of mappings verification we decided to implement a quiz like game similar to "Who wants to be a Millionaire" (with a small difference that instead of real questions we will mostly ask about words or phrases, composed of two or more words). One of the major problems here is how to order the questions asked? It is obvious that when the player starts the game, he or she shouldn't immediately get a difficult question – otherwise the player would leave the game quickly and would be discouraged. The difficulty should start at a relatively low level and increase while the player answers next questions. For this we need to somehow order the words from the least to the most difficult. In this paper we present a method of validating word difficulty for such ordering.

The structure of the paper is as follows. Section 2 defines what is a word difficulty and presents some of the approaches. Section 3 presents the proposed approach. In Sect. 4 evaluation of the proposed algorithm is given and finally Sect. 5 presents final conclusions and proposes some works that can be done in the future to further enhance the proposed solution.

2 Defining Word Difficulty and Related Work

To assess word difficulty we need an algorithmic way of classifying words having only that word, i.e. a set of tokens (letters in this case). Such sets are not really meaningful to a computer program in a way they are meaningful to an average person.

There are many approaches to this problem. Most of them however focus on assessing the whole text [10,11] and often utilize complex techniques like Coh-Metrix [12]. In our case single word, rather than the whole text, characterization is needed. One can also find plenty of games and puzzles that are related to words and they use various scoring systems and classification strategies. For example, Scrabble uses different score per letter depending on their frequency in specified language, promoting letters that are "rare" with higher score. In this case a difficult word might be a word that consist of rare letters. In this case the meaning or complexity of the word does not matter, only the score of letters it is composed of. On the other hand, a game in which the player's task is to type words on a standard QWERTY keyboard as fast as possible might not find these rules relevant – instead, a difficult word would be a one that is difficult to write, meaning for example its letters lay in some distance between each other or involving various fingers to type it. Therefore in order to correctly classify words, we need to define what words we want to consider difficult.

[2] https://play.google.com/store/apps/details?id=pl.gda.eti.kask.tgame, http://kask.eti.pg.gda.pl/tgame/.

There are many ways for people to tell if they consider a word difficult or not. It can be assumed that an average person knows an average set of words, so most of people will classify word's difficulties more or less similarly. Of course, it will be extremely unlikely that a group of people, asked to order the words by their difficulty, would end up with exactly same lineup. It's rather about asking them to assign some labels to the words, for example "how difficult, on a scale 1–5, is that word to you?". We can consider that a difficult word for an average person is a word that the person does not see often and, when they see that word, its construction gives them only a little, if any, idea about its origin or possible meaning. It is clear that a set of common, everyday words exists and have a large common part among average people. It is also clear, that an average person that is not in any way related to some specific domain might consider words coming from that domain as difficult, because those are not everyday words for them. In our case we thus define word difficulty as a score from 1 to 5 stating how probable it is that the person knows the meaning of the word.

Research show that frequency of a word might be correlated to its difficulty [13]. The observation that the words that occurs in texts less often can be considered more difficult is one of the best and most widely used methods of estimating word's difficulty. Such observations lead to creation of statistical word assessment defining difficulty as a measure how often the word occurs in the given domain or in everyday life meaning how easy it is to be seen. If the word is often seen in text than it is treated as easy for most of the population and vice versa [13,14]. Such approach led also to many modern text assessment service implementations like Twinword API [15].

3 Our Approach

In our solution we asses difficulty of words connected with Wikipedia articles. As such we decided to use Wikipedia as the corpora for text occurrence analysis. In all cases we evaluated a subset of the mappings against over 5 000 000 pages from English Wikipedia in each case building upon the previous solution.

3.1 Naive Approach

The first algorithm (Algorithm 1) was based directly on the observations given in the Sect. 2.

The result of the algorithm is a set of words **W** sorted from the easiest to the most difficult one. This approach have some limitations:

- a short, simple word that is very rare even among various types of texts can be mistakenly classified as a difficult word, for example a word "moo", which is clearly known even to very young children.
- a lot of words get the same score.
- the corpora used must have an average distribution of the words, otherwise it can yield incorrect results, this is difficult to verify with Wikipedia.

Algorithm 1. Naive approach

1: read text corpus **T**
2: read set of words to classify **W**
3: **for all** word **w** in set **W do**
4: count number of times **w** occurs in text **T**
5: assign the result **r** with **w**
6: **end for**
7: Sort words in **W** by number of occurrences **r** ascending

3.2 Adding Word Length

In this approach we decided to include into the algorithm the length of the analyzed word to reduce the impact of the first two limitations given in the previous section. In this approach we can consider difficult words as the ones that are both rare and long. Short words, as well as the frequent ones, will be more likely to be classified as easy.

During the implementation we also found out that it's very hard to correctly adjust the weight coefficients (length and number of occurrences) depending on the size of text corpora and input words, therefore we decided to use the relative values. The algorithm goes as shown in Algorithm 2.

Algorithm 2. The second approach

1: read text corpus **T**
2: read set of words to classify **W**
3: select the longest word in **W** and store its length as **lMax**
4: find the word from set W that occurs most times among text T and store the number of occurrences as **oMax**
5: **for all** word **w** in set **W do**
6: **c** = number of times **w** occurs in text **T**
7: **lw** = length of **w**
8: $s = (\frac{lw}{lMAX}) * (\frac{c}{oMAX})$
9: assign the score **s** with **w**
10: **end for**
11: Sort words in **W** by the score **s** descending

Once again the resulting set of words **W** is sorted from the easiest to the most difficult word. The score is calculated as a multiplication of length of the word and the number of occurrences in the text. In this approach the most difficult word (the lowest score) is a combination of being long and occur least times. If two words have similar frequency score, the longer one will now become more difficult. In all cases the score is calculated as relative within the available text.

3.3 The Final Approach

In this approach we tried to deal with the third issue. Imagine that our text set consist of ten texts, similar in length: five of them are academic papers on distributed

computations and five of them are cake recipes. If it happens that only one cake requires bananas, word "banana" might appear less times than the word "matrix" which would mean that the former will be classified as more difficult while one might expect that for an average person it would be vice versa. To eliminate this problem, we might take into consideration whether the word originates from an easy or a difficult text and apply proper weight to the resulting score. For determining whether a text is difficult or not (in English) we can use one of the existing methods. In our solution we chosen Flesch-Kincaid readability ease test [16]. This test, invented by Rudolf Flesch and J. Peter Kincaid for U.S. Navy in 1975, takes a text and applies formula 1 following to it.

$$fkScore = 206.835 - 1.015 * (\frac{totalwords}{totalsentences}) - 84.6 * (\frac{totalsyllables}{totalwords}) \quad (1)$$

The result (fkScore) is a score with most values between 0 and 100 (although scores below and above are possible to achieve). This score tells how difficult is the text to read – the lower the score, the more difficult the text. As a curiosity, one of the easiest sentences with score 116 is "The cat sat on the mat", whereas the chemical name of titin (a protein) which is 189,819 characters long and consists of 72443 syllables, scores a -6128472. The rated texts can be divided into groups show in Table 1.

Incorporation of Flesch-Kincaid score requires further changes to the algorithm. The final algorithm is shown in Algorithm 3.

As in previous approaches the resulting set of words **W** is sorted from the easiest to the most difficult word. Similar as in previous approach the final score is a multiplication of length and occurrence scores. In this case the length score of a word takes value from range <1, 2> so not to zero the whole score. In the case where classification of a multi-word phrase is needed the phrase should be split into separate words for which the score should be computed. The phrase score should be than computed as an average score of all words that it consists of.

Table 1. Groups of text difficulty

Score	Notes
90.0 and above	Very easy to read, easily understood by an 11-year old
80.0–90.0	Easy to read, conversational English for average people
70.0–80.0	Fairly easy to read
60.0–70.0	Plain English, easily understood by 13- to 15-year-old students
50.0–60.0	Fairly difficult to read
30.0–50.0	Difficult to read
30.0 and below	Very difficult to read

Algorithm 3. The third approach with Flesch-Kincaid score

read a set of texts **T**
read set of words to classify **W**
for i = 0 **to i** = size(**T**) **do**
    $fkScore[i] = 206.835 - 1.015 * (\frac{totalwords(T[i])}{totalsentences(T[i])}) - 84.6 * (\frac{totalsyllables(T[i])}{totalwords(T[i])})$
end for
for i = 0 **to i** = size(**T**) **do**
    $fkScore[i] = \frac{fkScore[i]}{max(fkScore)}$
end for
select the longest word in **W** and store its length as **lMax**
for i = 0 **to i** = size(**W**) **do**
    for j = 0 **to j** = size(**T**) **do**
        let **occurrences[i][j]** be the number of occurrences of word **W[i]** in text **T[j]**
        let **partialFrequencyScore[i][j]** be **occurrences[i][j]*fkScore[j]**
    end for
    $frequencyScore[i] = \frac{\sum_{n=0}^{size(T)-1} partialFrequencyScore[i][n]}{size(T)}$
    $lengthScore[i] = (2 - \frac{length(W[i])}{lMax})$
    $s[i] = frequencyScore[i] * lengthScore[i]$
end for
Sort words in **W** by the score **s** descending

4 Evaluation

For evaluation purposes we created a set of 20 words generated using Random Word Generator tool[3]. Those words were presented to a group of 90 people. Each time the words were randomly shuffled not to introduce any suggestions to the order of difficulty. Each person was asked to order the words from the easiest one to the most difficult. The results were than accumulated and merged using the following steps:

1. For each person participated, get a list of their result
2. For every list, attach scores with every word, so that the first word (the easiest, according to the participant) gets 1 point and the last (the most difficult) gets 20 points
3. For every word then calculate average score from all the lists.
4. Sort words by calculated average score ascending.

In the results we got an ordered list of words (from the easiest to the most difficult) presented in Table 2.

Just like we assumed, the common, well-known and short words occupy the beginning of the table. Some results might be considered unexpected (like "whistle"), but in those cases there might be other factors influencing their difficulty except for the ones that were considered in the formula (e.g. difference between how a word is written and pronounced – something that is common for English language).

[3] https://randomwordgenerator.com/.

After the survey the same set of words were classified using all three versions of the proposed solution. As text corpora, English Wikipedia was used (consisting of 4 838 000 pages).

We divided the results into four groups with equal number of words. We assume that words in one groups are close in terms of difficulty. The words in italics are the words that were assigned by the algorithm to other difficulty level than according to the survey. The results, compared with the survey, are presented in Table 3.

As we can see the naive version correctly assigned 9 words, first modification 10 words and the final version 11 words. The average difference of score (calculated in the number of levels the word was shifted) is 1.45 in the first case, 1.4

Table 2. Evaluation set of words ordered by humans

Word	Order	Word	Order	Word	Order	Word	Order
start	1	dirt	6	parcel	11	panoramic	16
cup	2	unit	7	jazzy	12	whistle	17
hard	3	rabbits	8	disagree	13	substantial	18
false	4	haircut	9	horrible	14	wobble	19
cute	5	drip	10	observation	15	tedious	20

Table 3. Comparison of words evaluation

Survey	Algorithm #1	Algorithm #2	Algorithm #3
start	cup	cup	cup
cup	start	start	*unit*
hard	*unit*	*unit*	hard
false	hard	hard	*drip*
cute	false	false	start
dirt	*substantial*	dirt	dirt
unit	*observation*	*substantial*	*wobble*
rabbits	dirt	*observation*	*false*
haircut	*whistle*	*whistle*	rabbits
drip	*parcel*	*cute*	*cute*
parcel	*rabbits*	parcel	jazzy
jazzy	*panoramic*	*rabbits*	*whistle*
disagree	horrible	horrible	parcel
horrible	*cute*	*panoramic*	*haircut*
observation	disagree	disagree	horrible
panoramic	*jazzy*	*drip*	*observation*
whistle	tedious	*jazzy*	tedious
substantial	*drip*	tedious	*disagree*
wobble	*haircut*	*haircut*	panoramic
tedious	wobble	wobble	substantial

in the second case and 0.9 for the final algorithm. The final solution, in most cases, switch the words between the neighboring groups.

The results obtained using the third approach suits our purposes very well. We needed to obtain an ordered list of words difficulty for the need of a quiz game. This list can be later corrected by the players themselves during the actual playing of the game. Moving words on level up or down in terms of difficulty is perfectly fine in this situation. For the purpose of the game we plan on classifying 56 000 words and assign them to 10 classes. Small differences with resulting positions are thus less likely to cause level mismatches.

The experiments performed during the evaluation revealed some unsolved problems. It was very hard to create an absolute scoring system within the algorithm. This is because the average frequency of a word in all texts depends on the number of text we use, while the average length of a word is independent of that and stabilizes at some level. This attempted to be fixed by introducing the weight coefficients for both the frequency and length parts, but it was very difficult to adjust them properly, especially when the text corpora contained millions of texts.

Flesch-Kincaid score works well for regular English texts, but sometimes within the corpora there are texts resulting in very low, negative scores, breaking the results. Such situation occurred in one of the tests. Some Wikipedia pages consists mostly of Chinese people names, which scored -2971. Such a text was not useful in our computations and introduced noise in the results. After some further tests we decided to exclude any text with a score lower than 0, since this indicated that either the text is not in English (at least partially) or we deal with something unusual, like a protein name.

The final approach is also very memory intensive. Counting occurrences of 56 000 unique words in over 4 800 000 texts takes a lot of time and uses a lot of memory. The earliest plan was to count occurrences of all of the unique words in all of the texts, but this led to exponential growth of the database. We thus decided to limit calculations only to the words we are actually interested in which allowed to keep the memory constraints in check.

5 Conclusion and Future Work

The scoring of word difficulty is a difficult task, mainly due the lack of one, standard definition what it means for a word to be difficult. Observation shows however, that the approach presented here is consistent with observation of the human behavior – if we stumble upon a word very often it is more probable that it is easier for us.

The presented approach allows us to order the set o phrases needed verification in a manner that will not be discouraging to players. Also it is worth noting that the main purpose of this algorithm is to give a preliminary classification that later might be corrected by players playing the game or manually adjusted by moderators.

In the future it could be feasible to extend the algorithm further. We should take into consideration also the pronunciation of the word. The greater the difference between spelling and pronunciation, the more difficult the word can be considered. Taking into account Scrabble-like letter ranking like promoting rare letter groups (pairs or triplets) or calculate the ratio of unique letters to the word length (more unique letters, the more difficult the word) might allow better classification independent from the domain of the word.

References

1. Korytkowski, R., Szymanski, J.: Collaborative approach to WordNet and Wikipedia integration. In: The Second International Conference on Advanced Collaborative Networks, Systems and Applications, COLLA, pp. 23–28 (2012)
2. Szymański, J.: Mining relations between Wikipedia categories. In: Zavoral, F., Yaghob, J., Pichappan, P., El-Qawasmeh, E. (eds.) NDT 2010. CCIS, vol. 88, pp. 248–255. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14306-9_25
3. Szymański, J.: Words context analysis for improvement of information retrieval. In: Nguyen, N.-T., Hoang, K., Jędrzejowicz, P. (eds.) ICCCI 2012. LNCS (LNAI), vol. 7653, pp. 318–325. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34630-9_33
4. Szymański, J., Duch, W.: Self organizing maps for visualization of categories. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012. LNCS, vol. 7663, pp. 160–167. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34475-6_20
5. Von Ahn, L.: Games with a purpose. Computer **39**(6), 92–94 (2006)
6. Von Ahn, L., Dabbish, L.: Designing games with a purpose. Commun. ACM **51**(8), 58–67 (2008)
7. Biuro Prasowe Samsung Electronics Polska Sp. z o.o.: Prawie połowa Polaków gra codziennie w gry wideo (in Polish)
8. Wightman, D.: Crowdsourcing human-based computation. In: Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, pp. 551–560. ACM (2010)
9. Boiński, T.: Game with a purpose for mappings verification. In: 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 405–409. IEEE (2016)
10. Binkley, M.R.: New ways of assessing text difficulty. Readability: Its past, present and future, pp. 98–120 (1988)
11. Kauchak, D., Leroy, G., Hogue, A.: Measuring text difficulty using parse-tree frequency. J. Assoc. Inf. Sci. Technol. **68**, 2088–2100 (2017)
12. Crossley, S.A., Greenfield, J., McNamara, D.S.: Assessing text readability using cognitively based indices. Tesol Q. **42**(3), 475–493 (2008)
13. Breland, H.M.: Word frequency and word difficulty: a comparison of counts in four corpora. Psychol. Sci. **7**(2), 96–99 (1996)
14. Carroll, J.B.: An alternative to Juilland's usage coefficient for lexical frequencies. ETS Res. Rep. Ser. **1970**(2), 1–15 (1970)

15. Inc., T.: Twinword API (2011). https://www.twinword.com/api/language-scoring.php. Accessed 10 May 2017
16. Kincaid, J.P., Fishburne Jr, R.P., Rogers, R.L., Chissom, B.S.: Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document (1975)