Category Robustness

If it looks like a duck...

Xavier Morin Duchesne
M. Sc. Computer Science
School of Computer Science, McGill University
Course: COMP762 – ML & NLP Methods for Software Engineering

Abstract— **When designing a tool dependent on human-defined categories, it is important to keep in mind that such categories are inherently vague and their boundaries ambiguous. We propose a novel measure, category robustness, which can be used to evaluate the semantic boundaries between categories. We demonstrate that this novel measure is correlated with logical proxies for semantic closeness and demonstrate a possible use case for this novel measure.**

I. Introduction

If it looks like a duck, swims like a duck, and quacks like a duck, then it must be a duck! But what about the platypus? It shares some of the most defining physical characteristics of a duck, yet it is not a duck. Where, then, does one draw the line? When is a duck a duck and a platypus, a platypus?

Absurd as this example may seem, it highlights the inherent vagueness of categories. As Andersen put it "semantic domains or 'fields', as well as specific items within them, in a given language often seem to defy any kind of adequate, unique, unambiguous description." [1] That is to say that, contrary to our intuition, categories are often vague and their boundaries fuzzy [2]. This vagueness and the lack of explicit semantic boundaries can in fact be shown to extend even to simple, common, everyday objects. Labov [3], for example, showed in his seminal work that the boundaries between common kitchenware such as cups, bowls, and vases, are fuzzy: Which physical features define a cup as a cup? More importantly, which physical features distinguish it from a bowl or vase? One could argue that the presence of a handle is defining of a cup, but a cup with a broken handle is still a cup. Similarly, size is an important yet insufficient factor: large cups and small bowls do exist.

These considerations—the inherent vagueness of real-world categories and the fuzziness of their semantic boundaries—are important to keep in mind when creating tools which depend on a set of human-defined categories. There are many instances in the Software Engineering literature [4]–[14] where a set of categories was first induced through manual inspection of a sample of a larger dataset, and where, then, a method was devised to assign new content to these categories, having been designed to reproduce the human intuition which engendered the categories in the first place. As argued earlier, however, human-defined categories, even those of simple, common, everyday objects, are inherently vague and lack well-defined semantic boundaries. This thus calls into question the quality of the categories proposed for such abstract concepts as developer communications [4]–[7], app reviews [8]–[13], or user feedback [14], and the validity of the related methods. Importantly, this also highlights the need for tools for evaluating the vagueness of those semantic categories and their boundaries.

Accordingly, the goal of this project was to evaluate a novel measure, category robustness, the purpose of which is to assess the semantic boundaries between categories. The idea behind category robustness is that it should be possible to exchange components of members belonging to semantically close categories without affecting category membership. To put it differently, exchanging parts between two cars yields two cars.[1] Likewise, exchanging parts between a car and a plane, though likely to yield two useless piles of scrap metal, may yield a (strange looking) car and plane. Conversely, exchanging parts between a car and a human being or, say, between a beaver and a duck, can only lead to some strange monstrosity, or, in the case of the beaver and duck, platypuses. By that logic, then, for a given pair of categories, category robustness is evaluated by exchanging components in members of the respective categories and evaluating the number the number of "platypuses" thereby created, in contrast to the number of new elements created which, despite the exchange, continue to fall neatly within one category or the other.

In what follows, we report our evaluation of category robustness on the set of intention mining categories proposed by Huang et al. [7]. More specifically, we created augmented sentences by exchanging noun phrases within the sentences in the Huang et al. dataset, and evaluate how often these resulting augmented sentences fall within either of their original categories and how often platypuses are created. These measures are correlated to measures other measures of category similarity described below and insights about the relationships between categories are discussed.

Please note that although we focus, here, on intention mining and, specifically, on the work done by Huang et al. [7], this approach is not specific to their dataset, nor is it specific to intention mining, and should be applicable to all areas of Software Engineering where categories have been proposed along with automated methods for assigning future content to those categories.

II. Related works

In his seminal work, Labov [3] investigated ambiguity in the "denotation of cups and cuplike containers, and the use of words such as <u>cup</u>, <u>bowl</u>, <u>dish</u>, <u>mug</u>, <u>glass</u>, <u>pitcher</u>, etc. and their corresponding terms in other languages."[2] He found that, when asked to name containers of various shapes,

[1]The resulting vehicles may not be functional or aesthetically pleasing, but we argue that they would no less be recognizable as cars.

[2]Text was underlined in the original work; the format was copied here for the sake of exactitude.

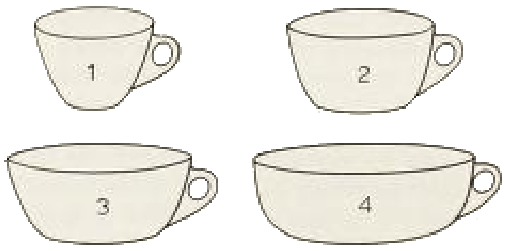

Fig. 1. Objects 1–4 from [3], Figure 5, page 354. In his seminal work, Labov investigated the impact of variations on width, depth, and shape on our judgement of whether a given object is a cup or bowl. Only variations on width are shown here.

widths, and depths, human observers gradually (rather than sharply) shifted their identifications from cup to bowl or to vase (see Figure 1 for examples of the containers used by Labov). In other words, human observers displayed fuzzy boundaries which resulted in gradual, rather than sharp, changes in describing cuplike containers. Labov ultimately concluded that function appeared to be the most important factor in determining the label assigned to cuplike containers: a vase is a vase, because it can contain flowers.

Andersen [1] followed up on this work by showing that young children first rely on physical features in order to name objects and that, in doing so, they tend to overextend words like "cup" to containers which an adult would name otherwise. As they age, however, they begin to rely less on perceptual properties and more strongly on functional properties, recognizing that categories are inherently vague and that, as stated above, a vase is a vase not necessarily by its inherent properties, but by the way that it is used. It is unclear however whether this recognition of the vagueness of categories and the fuzziness of semantic boundaries is domain-specific or generalizes to all categories.

Huang et al. [7] proposed a set of seven intention categories—aspect evaluation, feature request, information giving, information seeking, problem discovery, solution proposal, and meaningless[3]—and trained a Convolutional Neural Network (CNN) to classify sentences into those categories. The purpose of their work was to classify sentences in developer discussions in order to

[3]This category is mostly composed of messages thanking the developers for their work.

facilitate dissemination of the relevant information to the relevant people: bug reports to maintainers, feature requests to developers, praise to management, and so on. Their work was meant to improve upon the Di Sorbo et al. [4] set of six categories and corresponding linguistic patterns for assigning sentences to the categories. The Di Sorbo et al. work, in turn, was designed as an improvement over Guzzi et al. [6]. In all three cases, it should be noted, the categories were produced following manual inspection of the data. That is to say that the categories were not defined according to strict criteria, but, instead, according to human intuition which, as argued previously, is likely to produce vague categories with fuzzy boundaries. We use the freely available Huang et al. dataset for the purposes of this project.

Previous Natural Language Processing work interested in producing augmented sentences has occurred in other areas such as Grammatical Error Correction [15]. As far as we are aware, however, this work is the first to apply this kind of augmentation to the field of Software Engineering and more specifically to the analysis of boundaries between categories.

III. Methodology

A. Dataset

For the purposes of this project, we chose to use the data collected by by Huang et al. [7]. It comprised $6,400$ sentences: the 983 sentences collected by Di Sorbo et al. [4] and $5,417$ developer discussion sentences collected over four projects: Bootstrap [16], Docker [17], Tensorflow [18], and VSCode [19]. Huang et al. assigned each of these sentences to one of seven categories: aspect evaluation, feature request, information giving, information seeking, problem discovery, solution proposal, and meaningless. The dataset were split into a training set (90% of the data) and a validation set (10%).

B. Text preprocessing

Sentences were preprocessed using the Stanford Core NLP toolkit [20]; punctuation and stop words were removed, and the remaining words were lemmatized. Words were marked with a part-of-speech tag and with a tag corresponding to their most immediate phrase (e.g., noun phrases).

C. Machine learning models and analyses

Analyses and machine learning models were written in Python 3 [21], and ran on Google Colaboratory [22]. They relied mainly on three Python libraries: Numpy [23], Scikit-learn [24], and PyTorch [25].

Three machine learning models were used to evaluate category robustness: Support Vector Machines [26], a Long Short-Term Memory model [27], and the Huang et al. Convolutional Neural Network [7].

For the SVM model, a processing pipeline was created where sentences were first transformed to a bag-of-words representation, before being to a TF-IDF representation [28]. The TFIDF vectors corresponding to each sentence where finally fed to the Support Vector classifier which used a linear kernel with $C = 1$.

For the LSTM, words were passed through an embedding layer of size 128 before being passed to the LSTM itself. The LSTM had two layers with 256 hidden states each.

Finally, similarly to the LSTM, words first passed through an embedding layer of size 128 before being passed to the CNN itself. The CNN was composed of three parallel convolutional layers with filter sizes of 3, 4, and 5 respectively. The output of these layers was given to a dropout layer, followed by a rectified linear unit, before being concatenated and finally given to a linear classifier. See [7] for more details.

D. Term-based methods

Two term-based methods were used: TF-IDF [28] and In-Out scores.

TF-IDF or term frequency-inverse document frequency is a measure commonly used to identify important terms within a set of documents. It gives high scores to terms which occur frequently in few documents and low scores to terms which occur frequently across documents. This is meant, for example, to give low scores to common words such as the article “the” or the verb “to be.”

The In-Out score was created specifically for our purposes and it is defined as follows:

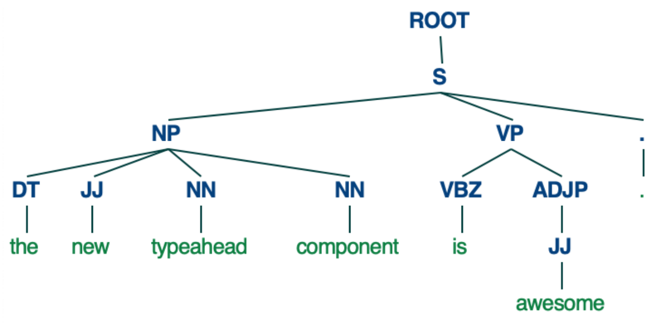

Fig. 2. Example of a parse tree.

		AE	FR	IG	IS	PD	SP	ML
TF-IDF	*POS*	N	N	N	N	N	N	N
	phrase	NP	NP	NP	NP	NP	NP	NP
In-Out	*POS*	J	J	N	N	N	N	J
	phrase	JP	JP	NP	NP	NP	NP	NP

TABLE I
THE PARTS OF SPEECH AND PHRASES WHICH PRESENTED THE HIGHEST TOTAL TFIDF SCORE AND HIGHEST TOTAL IN-OUT SCORE FOR EACH CATEGORY.
N = NOUN, J = ADJECTIVE, NP = NOUN PHRASE, JP = ADJECTIVE PHRASE

$$\begin{aligned}\mathrm{In-Out}_{\mathrm{W,C}} = \\ & z\left(\#\text{occurrences of } W \text{ in } C\right) \\ & -z\left(\#\text{occurrences of } W \text{ not in } C\right)\end{aligned}$$

where $z()$ calculates the z-score, and W and C are respectively the word and category of interest. Using this measure, we obtain a score for each word and category representing the importance of that word to that category.

E. Sentence augmentation

For the purposes of this report, augmented sentences were created using a new method: phrase swapping. In phrase swapping, phrases (e.g., noun phrases) in two different sentences are identified and swapped. For example, let us consider the following pair of sentences:

The new typeahead component is awesome.
We need a dark/inverted style alert for this.

where the first sentence is part of the "aspect evaluation" category and the second, "feature request." In each of these sentences, a noun phrase has been identified and underlined. We can swap these noun phrases across sentences to obtain:

A dark/inverted style alert is awesome.
We need the new typeahead component for this.

Programmatically, this is achieved using the Stanford Core NLP toolkit [20]. The toolkit is used to obtain a parse tree for each sentence (see Figure 2 for an example); it is then simply a question of identifying subtrees in each sentence with the appropriate chunk type and swapping them.

	AE	FR	IG	IS	PD	SP	ML
AE		.90	.91	.65	.73	.70	.60
FR			.83	.55	.61	.59	.52
IG				.87	.88	.83	.80
IS					.62	.55	.45
PD						.66	.50
SP							.47
ML							

TABLE II
CATEGORY ROBUSTNESS SCORES FOR EACH PAIR OF CATEGORIES AS CALCULATED BY THE SVM.

	AE	FR	IG	IS	PD	SP	ML
AE		.92	.89	.92	.86	.83	.88
FR			.83	.86	.76	.75	.82
IG				.91	.89	.81	.85
IS					.88	.82	.86
PD						.82	.86
SP							.80
ML							

TABLE III
CATEGORY ROBUSTNESS SCORES FOR EACH PAIR OF CATEGORIES AS CALCULATED BY THE LSTM.

	AE	FR	IG	IS	PD	SP	ML
AE		.89	.94	.94	.85	.80	.86
FR			.98	.83	.72	.72	.79
IG				.94	.93	.86	.87
IS					.84	.80	.81
PD						.79	.79
SP							.91
ML							

TABLE IV
CATEGORY ROBUSTNESS SCORES FOR EACH PAIR OF CATEGORIES AS CALCULATED BY THE HUANG ET AL. CNN [7]

	AE	FR	IG	IS	PD	SP	ML
AE		.53	.50	.47	.35	.40	.24
FR			.53	.50	.43	.44	.20
IG				.55	.47	.53	.19
IS					.45	.47	.26
PD						.50	.17
SP							.19
ML							

TABLE V

PROPORTION OF OVERLAP BETWEEN PAIRS OF CATEGORIES IN THE TOP-100 WORDS WITH THE HIGHEST TF-IDF SCORES.

	SVM	LSTM	CNN	TF-IDF	In-Out
SVM		.60	.40	.60	.28
LSTM			.85	.36	.32
CNN				.25	.27
TF-IDF					.44
In-Out					

TABLE VII

CORRELATIONS BETWEEN THE DIFFERENT MEASURES OF CATEGORY ROBUSTNESS AND CATEGORY OVERLAP.

	AE	FR	IG	IS	PD	SP	ML
AE		.08	.03	.03	.00	.02	.09
FR			.05	.04	.02	.04	.06
IG				.17	.05	.09	.01
IS					.04	.10	.06
PD						.05	.00
SP							.00
ML							

TABLE VI

PROPORTION OF OVERLAP BETWEEN PAIRS OF CATEGORIES IN THE TOP-100 WORDS WITH THE HIGHEST IN-OUT SCORES.

IV. RESULTS AND DISCUSSION

One of the pitfalls of exploratory studies like this one is that there are often a large number of directions that the study can take with little-to-no guidance on which direction it should take. In our case, for example, swapping noun phrases seemed like an obvious choice, but swapping verb phrases and substituting verbs also seemed interesting. Unfortunately, swapping verb phrases is more complex: one verb may be transitive and the other intransitive; one verb may be conjugated for a third person pronoun (e.g., he has) whereas the other is conjugated for a first pronoun (I find). It was thus important for us to investigate further before investing time and effort into creating heuristics which may turn out not to be useful.

Our first step was therefore to use term-based methods, TF-IDF and In-Out scores, in order to identify the most important parts-of-speech and phrases in our dataset. In order to do so, each category's sentences into a single document and calculated a TF-IDF score for each word in those documents. We similarly calculated an In-Out score for word and category. We then calculated part-of-speech and phrase scores by summing the scores of words assigned with the part-of-speech or part of the phrase type. The results, presented in Table I, suggest that nouns and noun phrases were the most important part-of-speech and phrase, respectively. We accordingly focus on noun phrases in what follows.

Our second step was to create augmented sentences by swapping noun-phrases between sentences of the same and different categories, and to use the three models, SVM, LSTM, and CNN, trained beforehand on the Huang et al. [7] data, to assign categories to the new augmented sentences. As discussed earlier, the idea is that exchanges between semantically close categories are more likely to produce augmented sentences which still fit in those categories (i.e., a duck or a beaver), whereas exchanges between semantically far categories (robust categories) are more likely to create sentences which fit in neither (i.e., platypuses). The results are displayed in Tables II, III, and IV for the SVM, LSTM, and CNN respectively.

Having now evaluated category robustness using three machine learning models, we wanted to confirm that category robustness was actually getting at something, some inherent properties of those categories. In order to do so, we calculated overlap scores for each category pair using both TF-IDF and In-Out. That is to say that, for each category, we obtained the top 100 words with the highest TF-IDF and In-Out scores and then counted the number of words shared between pairs of categories. Both TF-IDF and In-Out scores are meant to represent the importance of words; if a given word is important to two categories, this suggests that those categories are closer than if

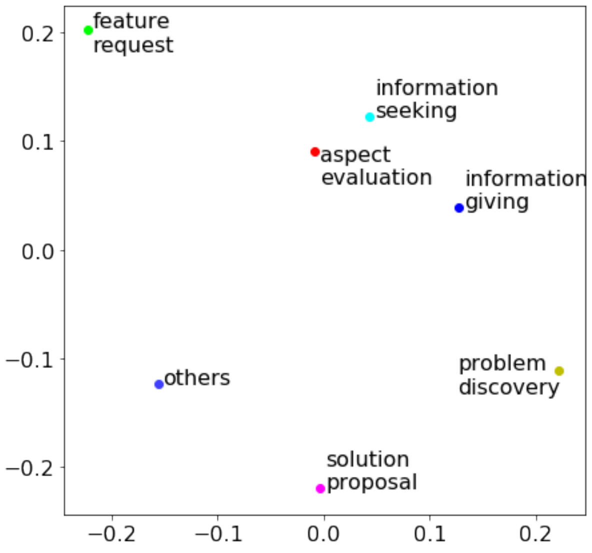

Fig. 3. The relationships between categories as ascertained by the Huang et al. [7] CNN, represented in two dimensions using Multidimensional Scaling. Categories that are close in this graph were found to be close semantically as well.

that that word were not important to both. Therefore, by measuring this overlap, we can get an idea of the semantic distance between categories. The results are presented in Tables V and VI. Of note, the In-Out overlap scores are far lower than those for TF-IDF. This is not surprising: the In-Out measure was designed explicitly to identify words that are important to a category, but not to to others. Having obtained measures of word overlap, we compared all of the available measures to each other and display the results in Table VII. Interestingly, all correlations are greater than zero and would qualify, at the very least, as small correlations (between .2 and .3), with some correlations which would even qualify as large ($.60$ and 0.85). This suggests that category robustness does embody a property of categories related to the ambiguity of semantic boundaries.

Finally, it is worth mentioning that Multidimensional Scaling**mds** can be applied to category robustness scores (for example those of the CNN, reported in Table IV) in order to obtain a visualization of the relationship between categories (like the one in Figure 3). In a such a graph, points that are close together represent categories that are semantically closer, whereas points that are far away represent categories that more mutually robust.

V. Conclusion

Human-defined categories are inherently vague and their boundaries fuzzy. We set out to explore a novel measure, category robustness, for evaluating the boundaries between categories. We found that our measure correlates well with other, commonsense measures of similarity between categories. Finally, we combine our approach with Multidimensional Scaling to get insights into the relationships between categories.

References

[1] E. S. Andersen, “Cups and glasses: Learning that boundaries are vague,” *Journal of Child Language*, vol. 2, no. 1, pp. 79–103, 1975.

[2] A. Lehrer, “Indeterminacy in semantic description,” *Glossa*, vol. 4, no. 1, pp. 87–110, 1970.

[3] W. Labov, “The boundaries of words and their meaning,” in *New Ways of Analyzing Variation in English*, C.-J. N. Bailey and R. W. Shuy, Eds., Georgetown University: Southeastern Conference on Linguistics, 1972, pp. 340–371.

[4] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. Gall, “Deca: Development emails content analyzer,” in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, IEEE, 2016, pp. 641–644.

[5] A. Guzzi, A. Begel, J. K. Miller, and K. Nareddy, “Facilitating enterprise software developer communication with cares,” in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, IEEE, 2012, pp. 527–536.

[6] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. v. Deursen, “Communication in open source software development mailing lists,” in *Proceedings of the 10th Working Conference on Mining Software Repositories*, IEEE Press, 2013, pp. 277–286.

[7] Q. Huang, X. Xia, D. Lo, and G. C. Murphy, “Automating intention mining,” *IEEE Transactions on Software Engineering*, 2018.

[8] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM, 2016, pp. 499–510.

[9] X. Gu and S. Kim, ""what parts of your apps are loved by users?"(t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2015, pp. 760–770.

[10] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*, IEEE, 2015, pp. 116–125.

[11] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proceedings of the 39th international conference on software engineering*, IEEE Press, 2017, pp. 106–117.

[12] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*, IEEE, 2015, pp. 281–290.

[13] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, IEEE, 2016, pp. 14–24.

[14] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*, IEEE, 2013, pp. 125–134.

[15] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. Inui, "An empirical study of incorporating pseudo data into grammatical error correction," *ArXiv preprint arXiv:1909.00502*, 2019.

[16] *Bootstrap*, https://github.com/twbs/bootstrap.

[17] *Docker*, https://github.com/moby/moby.

[18] *Tensorflow*, https://github.com/tensorflow/tensorflow.

[19] *VSCode*, https://github.com/Microsoft/vscode.

[20] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010.

[21] G. Van Rossum and F. L. Drake, *Python 3 reference manual*. Paramount, CA: CreateSpace, 2009, ISBN: 1441412697, 9781441412690.

[22] E. Bisong, "Google colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, 2019, pp. 59–64.

[23] T. E. Oliphant, *Guide to numpy*, 2nd. USA: CreateSpace Independent Publishing Platform, 2015, ISBN: 151730007X, 9781517300074.

[24] 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, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[26] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.