Query Classification using Wikipedia's Category Graph

Milad AlemZadeh
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, Ontario, Canada
Email: malemzad@uwaterloo.ca

Richard Khoury
Department of Software Engineering, Lakehead University, Thunder Bay, Ontario, Canada,
Email: richard.khoury@lakeheadu.ca

Fakhri Karray
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, Ontario, Canada
Email: karray@uwaterloo.ca

Abstract— **Wikipedia's category graph is a network of 300,000 interconnected category labels, and can be a powerful resource for many classification tasks. However, its size and the lack of order can make it difficult to navigate. In this paper, we present a new algorithm to efficiently exploit this graph and accurately rank classification labels given user-specified keywords. We highlight multiple possible variations of this algorithm, and study the impact of these variations on the classification results in order to determine the optimal way to exploit the category graph. We implement our algorithm as the core of a query classification system and demonstrate its reliability using the KDD CUP 2005 and TREC 2007 competitions as benchmarks.**

Index Terms—**Keyword search, Natural language processing, Knowledge based systems, Web sites, Semantic Web**

I. Introduction

Query classification is the task of Natural Language Processing (NLP) whose goal is to identify the category label, in a predefined set, that best represents the domain of a question being asked. An accurate query classification system would be beneficial in many practical systems, including search engines and question-answering systems. Query classification shares some similarities with other categorization tasks in NLP, and with document classification in particular. However, the challenge of query classification is accentuated by the fact that a typical query is only between one and four words long [1], [2], rather than the hundreds or thousands of words one can get from an average text document. Such a limited number of keywords makes it difficult to select the correct category label, and moreover it makes the selection very sensitive to "noise words", or words unrelated to the query that the user entered for some reason such as because they didn't remember a correct name or technical term to query for. A second challenge of query classification comes from the fact that, while document libraries and databases can be specialized to a single domain, the users of query systems expect to be able to ask queries about any domain at all [1].

This paper continues our work on query classification using the Wikipedia category graph [3], [4]. It refines and expands on our previous work by studying multiple different design alternatives that similar classification systems could opt for, and considers the impact of each one. In contrast with our previous papers, the focus here is not on presenting a single classification system, but on implementing and comparing multiple systems that differ on critical points.

The rest of the paper is organized as follows. Section 2 presents overviews of the literature in the field of query classification with a special focus on the use of Wikipedia for that task. We present in detail our ranking and classification algorithm in Section 3, and take care to highlight the points where we considered different design options. Each of these options was implemented and tested, and in Section 4 we describe and analyze the experimental results we obtained with each variation of our system. Finally, we give some concluding remarks in Section 5

II. Background

Query classification is the task of NLP that focuses on inferring the domain information surrounding user-written queries, and on assigning to each query the best category label from a predefined set. Given the ubiquity of search engines and question-handling systems today, this challenge has been receiving a growing amount of attention. For example, it was the topic of the ACM's annual KDD CUP competition in 2005 [5], where 37 systems competed to classify a set of 800,000 real web queries into a set of 67 categories designed to cover most topics found on the internet. The winning system was designed to classify a query by comparing its word vector to that of each website in a set pre-classified in the

Google directory. The query was assigned the category of the most similar website, and the directory's set of categories was mapped to the KDD CUP's set [2]. This system was later improved by introducing a bridging classifier and an intermediate-level category taxonomy [6].

Most query classifiers in the literature, like the system described above, are based on the idea of mapping the queries into an external knowledge source (an objective third-party knowledge base) or internal knowledge source (user-specific information) to classify them. This simple idea leads to a great variety of classification systems. Using an internal knowledge source, Cao *et al.* [7] developed a query classifier that disambiguates the queries based on the context of the user's recent online history. And on the other hand, many very different knowledge sources have been used in practice, including ontologies [8], websites [9], web query logs [10], and Wikipedia [4], [11], [12].

Exploiting Wikipedia as a knowledge source has become commonplace in scientific research. Several hundreds of journal and conference papers have been

classifier, but its fundamental principles differ fundamentally from [12]. Instead of using titles and articles to pinpoint the categories in which to classify a query like was done in [12], the classifier of [4] used titles only to create a set of inexact initial categories for the query and then explored the category graph to discover the best goal categories from a set of predetermined valid classification goals. This classifier also differs from the one described in this work on a number of points, including the equations used to weight and rank categories and the mapping of the classification goals. But the most fundamental difference is the use in this paper of pre-computed base-goal category distances instead of an exploration algorithm. As we will show in this paper, all these modifications are justified both from a theoretical standpoint and practically by improvements in the experimental results.

While using Wikipedia for query classification has not been a common task, there have been several document classification projects done using that resource which are worth mentioning. Schönhofen [13] successfully developed a complete document classifier using hundreds of journal and conference papers have been published using this tool since its creation in 2001. However, while both query classification and NLP using Wikipedia are common challenges, to the best of our knowledge there have been only three query classification systems based on Wikipedia.

The first of these three systems was proposed by Hu et al. [11]. Their system begins with a set of seed concepts to recognize, and it retrieves the Wikipedia articles and categories relevant to these concepts. It then builds a domain graph by following the links in these articles using a Markov random walk algorithm. Each step from one concept to the next on the graph is assigned a transition probability, and these probabilities are then used to compute the likelihood of each domain. Once the knowledge base has been build in this way, a new user query can be classified simply by using its keywords to retrieve a list of relevant Wikipedia domains, and sorting them by likelihood. Unfortunately, their system remained small-scale and limited to only three basic domains, namely “travel”, “personal name” and “job”. It is not a general-domain classifier such as the one we aim to create.

The second query classification system was designed by one of our co-authors in [12]. It follows Wikipedia's encyclopedia structure to classify queries step-by-step, using the query's words to select titles, then selecting articles based on these titles, then categories from the articles. At each step, the weights of the selected elements are computed based on the relevant elements in the previous step: a title's weight depends on the words that selected it, an article's weight on the titles', and a category's weight on the articles'. Unlike [11], this system was a general classifier that could handle queries from any domain, and its performance would have ranked near the top of the KDD CUP 2005 competition.

The last query classification system is our own previous work, described in [4]. It is also a general

developed a complete document classifier using Wikipedia, by mapping the document's vocabulary to titles, articles, and finally categories, and weighting the mapping at each step. In fact, we used some of the mapping techniques he developed in one of our previous works [12]. Alternatively, other authors use Wikipedia to enrich existing text classifiers by improving upon the simple bag-of-words approach. The authors of [14] use it to build a kernel to map the document's words to the Wikipedia article space and classify there, while the authors of [15] and [16] use it for text enrichment, to expand the vocabulary of the text by adding relevant synonyms taken from Wikipedia titles. Interestingly, improvements are reported in the classification results of [13], [15] and [16], while only [14] reports worse results than the bag-of-words method. The conclusion seems to be that working in the word space is the better option; a conclusion that [14] also shares. Likewise, that is the approach we used in the system we present in this paper.

III. Algorithm

Wikipedia's category graph is a massive set of almost 300,000 category labels, describing every domain of knowledge and ranging from the very precise, such as “fictional secret agent and spies”, to the very general, such as “information”. The categories are connected by hypernym relationships, with a child category having an “is-a” relationship to its parents. However, the graph is not strictly hierarchic: there exist shortcuts in the connections (i.e. starting from one child category and going up two different paths of different lengths to reach the same parent category) as well as loops (i.e. starting from one child category and going up a path to reach the same child category again).

The query classification algorithm we propose in this paper is designed to exploit this graph structure. As we will show in this section, it is a three-stage algorithm, with a lot of flexibility possible within each step. The first

```
Input: Wikipedia database dump
1.  CG ← the Category Graph
    extracted from Wikipedia
2.  Associate to each category in CG
    the list of all titles pointing
    to it
3.  GC ← the set of Goal Categories
    identified in CG
4.  Dist(GC,CG) ← the shortest-path
    distance between every GC and all
    categories in CG
Input:  User query, CG
5.  KL ← Keyword List of all
    keywords in the user query
6.  TL ← Title List of all titles in
    CG featuring at least one word in
    KL
7.  KTW ← Keyword-Title Weight, a
    list containing the weight of a
    keyword from KL featured in a
    title from TL
8.  BC ← Base Categories, all
    categories in CG pointed to by TL
```

names that may refer to it). For example, the article for the United States is under the main title “United States”, as well as the redirect titles “USA”, “United States of America” and “United Staets” (common typo redirection), and the disambiguation title “America”. Our pre-processing deletes stopwords and punctuation from the titles, then maps them directly to the categories of the articles and discards the articles. After this processing, we find that our category graph features 5,453,808 titles and 282,271 categories.

The next step in the graph construction category is to define a set of goal categories that are acceptable classification labels. The exact number and nature of these goal categories will be application-specific. However, the set of Wikipedia category labels is large enough to cover numerous domains at many levels of precision, which means that it will be easy for system designers to identify a subset of relevant categories for their applications, or to map an existing category set to Wikipedia categories.

The final pre-processing step is to define, compute and store the distance between the goal categories and every category in the graph. This distance between two

```
9.  CD ← Category Density for all BC
    computed from the KTW
10. BC ← top BC ranked by CD
Input: GC, DIST(GC,BC), CD
11. GS ← Goal Score of each GC,
    computed based on their distance
    to each BC and on CD
12. Return: top 3 GC ranked by GS
```

Figure 1. Structure of the three steps of our classification algorithm: the pre-processing step (top), the base category evaluation (middle), and the exploration for the goal categories (bottom).

stage is a pre-processing stage, during which the category graph is built and critical application-specific information is determined. This stage needs to be done only once to create the system, by contrast with the next two stages that are executed for each submitted query. In the second stage, a user's query is mapped to a set of base categories, and these base categories are weighted and ranked. And finally, the algorithm explores the graph starting from the base categories and going towards the nearest goal categories in stage 3. The pseudocode of our new algorithm is shown in Figure 1.

A. *Stage 1: Pre-Processing the Category Graph*

We begin the first stage of our algorithm by extracting the list of categories in Wikipedia and the connections between categories from the database dump made freely available by the Wikimedia Foundation. For this project, we used the version available from September 2008.

Furthermore, our graph includes one extra piece of information in addition to the categories, namely the article titles. In Wikipedia, each article is an encyclopedic entry on a given topic which is classified in a set of categories, and which is pointed to by a number of titles: a single main title, some redirect titles (for common alternative names, including foreign translations and typos) and some disambiguation titles (for ambiguous

categories is the number of intermediate categories that must be visited on the shortest path between them. We allow any path between two categories, regardless of whether it goes up to parent categories or down to children categories or zigzags through the graph. This stands in contrast with our previous work [4], where we only allowed paths going from child to parent category. The reason for adopting this more permissive approach is to make our classifier more general: the parent-only approach may work well in the case of [4] where all the goal categories selected were higher in the hierarchy than the average base category, but it would fail miserably in the opposite scenario when the base categories are parents of the goal categories. When searching for the shortest paths, we can avoid the graph problems we mentioned previously, of multiple paths and loops between categories, by only saving the first encounter of a category and by terminating paths that are revisiting categories. Finally, we can note that, while exploring the graph to find the shortest distance from every goal category and all other categories may seem like a daunting task, for a set of about 100 goal queries such as we used in our experiments it can be done in only a few minutes on a regular computer.

B. *Stage 2: Discovering the Base Categories*

The second stage of our algorithm as shown in Figure 1 is to map the user's query to an initial set of weighted base categories. This is accomplished by stripping the query of stopwords to keep only relevant keywords, and then generating the exhaustive list of titles that feature at least one of these keywords. Next, the algorithm considers each title *t* and determines the weight W_t of the keywords it contains. This weight is computed based on two parameters: the number of keywords featured in the title (N_k), and the proportional importance of keywords in the title (P_k). The form of the weight equation is given in

equation (1).

$$W_t = N_k P_k \tag{1}$$

The number of keywords featured in the title is a simple and unambiguous measure. The proportional importance is open to interpretation however. In this research, we considered three different measures of importance. The first is simply the proportion of keywords in the title (N_k / N_t, where N_t is the total number of words in title *t*). The second is the proportion of characters in the title that belong to keywords (C_k / C_t, where C_k is the number of characters of the keywords featured in the title and C_t is the total number of characters in title *t*). This metric assumes that longer keywords are more important; in the context of queries, which are only a few words long [1], [2], it may be true that more emphasis was meant by the user on the longest, most evident word in the query. The final measure of proportional importance is based on the word's inverted frequencies. It is computed as the sum of inverted frequencies of the keywords in the title to the sum of frequencies of all title words (ΣF_k / ΣF_t), where the inverted frequency of a word *w* is computed as:

$$F_w = \ln(T / T_w) \tag{2}$$

the number of query keywords. It happens in the case where each keyword has its maximum value in that category, meaning that one of the titles pointing to the category is composed of exactly the query words.

$$D_i = \sum_k \max_t (W_t^{k,i}) \tag{3}$$

At the end of this stage of the algorithm, we have a weighted list of base categories, featuring some categories pointed to by high-weight words and summing to a high density score, and a lot of categories pointed to by only lower-weight words and having a lower score. In our experiments, we found that the set contains over 3,000 base categories on average. We limit the size of this list by keeping only the set of highest-density categories, as categories with a density too low are deemed to be too unrelated to the original query to be of use. This can be done either on a density basis (i.e. keeping categories whose density is more than a certain proportion of the highest density obtained for this query, regardless of the number of categories this represents, as we did in [4]) or on a set-size basis (i.e. keeping a fixed number of categories regardless of their density, the approach we will prioritize in this paper). When using the set-size approach, a question arises on how to deal with ties when the number of tied categories exceeds the size of the set to

In equation (2), T is the total number of titles in our category graph and T_w is the number of titles featuring word w. It is, in essence, the IDF part of the classic term frequency-inverse document frequency (TFIDF) equation: $(N_w / N)\ln(T / T_w)$, where N_w is the number of instances of word w in a specific title (or more generally, a document) and N is the total number of words in that title. The TF part (N_w / N) is ignored because it does not give a reliable result when dealing with short titles that only feature each word once or twice. We have used this metric successfully in the past in another classifier we designed [12].

We can see from equation (1) that every keyword appearing in a title will receive the same weight W_t. Moreover, when a title is composed exclusively of query keywords, their weight will be the number of keywords contained in the title. The maximum weight a keyword can have is thus equal to the number of keywords in the query; it occurs in the case where a title is composed of all query keywords and nothing else.

Next, our algorithm builds a set of base categories by listing exhaustively all categories pointed to by the list of titles. This set of base categories can be seen as an initial coarse classification for the query. These base categories are each assigned a density value. A category's density value is computed by determining the maximum weight each query keyword takes in the list of titles that point to that category, then summing the weights of all keywords, as shown in equation (3). In that equation, D_i is the density of category i, and $W_t^{k,i}$ refers to the weight W_t of a title t that contains keyword k and points to category i. Following our discussion on equation (1), we can see that the maximum density a category can have is the square of

return. In our system, we break ties by keeping a count of the number of titles that feature keywords and that point to each category, and giving priority to the categories pointed to by more titles.

C. *Stage 3: Ranking the Goal Categories*

Once the list of base categories is available, the third and final stage of the algorithm is to determine which ones of the goal categories identified in the first stage are the best classification labels for the query. As we outlined in the pseudocode of Figure 1, our system does this by ranking the goal categories based on their shortest-path distance to the selected base categories. There are of course other options that have been considered in the literature. For example, Coursey and Mihalcea [17] proposed an alternative metric based on graph centrality, while Syed *et al.* [18] developed a spreading activation scheme to discover related concepts in a set of documents. Some of these ideas could be adapted into our method in future research.

However, even after settling on the shortest-path distance metric, there are many ways we could take into account the base categories' densities into the goal categories' ranking. The simplest option is to use it at a threshold value – to cut off base categories that have a density lower than a certain value, and then rank the goal categories according to which are closest to any remaining base category regardless of density. That is the approach we used in [4]. On the other hand, taking the density into account creates different conditions for the system. Since some base categories are now more important than others, it becomes acceptable, for example, to rank a goal that is further away from several high-density base categories higher than a goal that is

closer to a low-density base category. We thus define a ranking score for the goal categories, as the sum for all base categories of a ratio of their density to the distance separating the goal and base. There are several ways to compute this ratio; five options that we considered in this study are:

$$S_j = \sum_i D_i / (dist(i,j) + 0.0001) \quad (4)$$

$$S_j = \sum_i D_i / (dist(i,j)^2 + 0.0001) \quad (5)$$

$$S_j = \sum_i D_i e^{-dist(i,j)} \quad (6)$$

$$S_j = \sum_i D_i e^{-2dist(i,j)} \quad (7)$$

$$S_j = \sum_i D_i e^{-dist(i,j)^2} \quad (8)$$

In each of these equations, the score S_j of goal category j is computed as the sum, for all base categories i, of the density D_i of that category, which was computed in equation (3), divided by a function of the distance between categories i and j. This function is a simple division in equations (4) and (5), but the exponential in equations (6-8) put progressively more importance on the distance compared to the density. The addition of 0.0001 in equations (4) and (5) is simply to avoid a division by

Precision Award was given to the system with the top overall precision value within the top 10 systems evaluated on overall F1 value. Overall Recall was not used in the competition, but is included here because it is useful in our experiments.

$$\text{Precision} = \frac{\sum_j \text{queries correctly labeled as } c_j}{\sum_j \text{queries labeled as } c_j} \quad (9)$$

$$\text{Recall} = \frac{\sum_j \text{queries correctly labeled as } c_j}{\sum_j \text{queries belonging to } c_j} \quad (10)$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

$$\text{Overall Precision} = \frac{1}{3}\sum_{L=1}^{3} \text{Precision against labeler } L \quad (12)$$

$$\text{Overall Recall} = \frac{1}{3}\sum_{L=1}^{3} \text{Recall against labeler } L \quad (13)$$

$$\text{Overall F1} = \frac{1}{3}\sum_{L=1}^{3} \text{F1 against labeler } L \quad (14)$$

In order for our system to compare to the KDD CUP competition results, we need to use the same set of category labels. As we mentioned in Section 3, the size and level of detail of Wikipedia's category graph makes it possible to identify categories to map most sets of labels in equations (4) and (5) is simply to avoid a division by zero in the case where a selected base category is also a goal category.

Finally, the goal categories with the highest score are returned as classification results. In our current version of the system, we return the top three categories, to allow for queries to belong to several different categories. We believe that this corresponds to a human level of categorization; for example, in the KDD CUP 2005 competition [5], human labelers used on average 3.3 categories per query. However, this parameter is flexible, and we ran experiments keeping anywhere from one to five goal categories.

IV. Experimental Results

The various alternatives and options for our classifier described in the previous section were all implemented and tested, in order to study the behavior of the system and determine the optimal combination. That optimal combination was then subjected to a final set of tests with new data.

In order to compare and study the variations of our system, we submitted them all to the same challenge as the KDD CUP 2005 competition [5]. The 37 solutions entered in that competition were evaluated by classifying a set of 800 queries into up to five categories from a predefined set of 67 target categories c_j and comparing the results to the classification done by three human labelers. The solutions were ranked based on overall precision and overall F1 value, as computed by Equations (9-14). The competition's Performance Award was given to the system with the top overall F1 value, and the

possible to identify categories to map most sets of labels to. In our case, we identified 99 goal categories in Wikipedia corresponding to the 67 KDD CUP category set. These correspondences are presented in Appendix A.

A. Proportional Importance of Keywords

The first aspect of the system we studied is the different formulae for the proportional importance of query keywords in a title. As we explained in Section IIIB, the choice of formula has a direct impact on the system, as it determines which titles are more relevant given the user's query. This in turn determines the relevance of the base categories that lead to the goal categories. A bad choice at this stage can have an impact on the rest of the system.

The weight of a title, and of the query keywords it contains, is function of the two parameters presented in equation (1), namely the number of keywords present in the title and the importance of those keywords in that title. Section IIIB gives three possible mathematical definitions of keyword importance in a title. They are a straightforward proportion of keywords in the title, the proportion of characters in the title that belong to keywords, and the proportion of IDF of keywords to the total IDF of the title, as computed with equation (2). We implemented all three equations and tested the system independently using each. In all implementations, we limited the list of base categories to 25, weighted the goal categories using equation (5), and varied the number of returned goal categories from 1 to 5.

The results of these experiments are presented in Figure 2. The three different experiments are shown with different grey shades and markers: dark squares for the

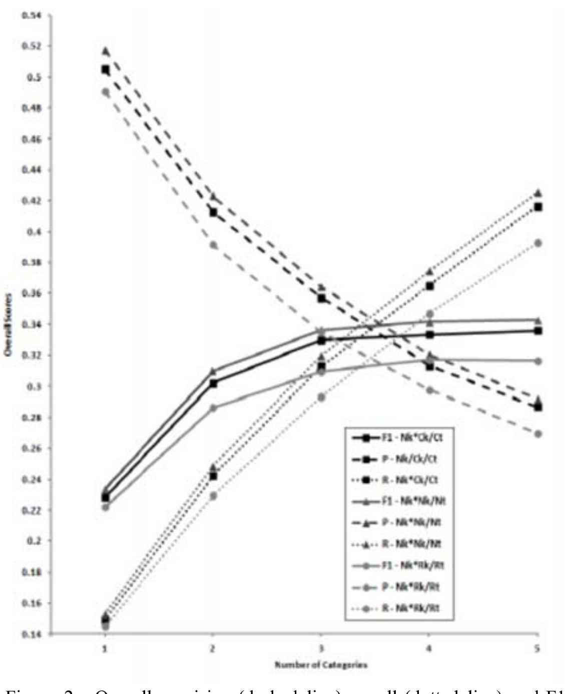

Figure 2. Overall precision (dashed line), recall (dotted line) and F1 (solid line) using $N_k*(C_k/C_t)$ (dark squares), $N_k*(N_k/N_t)$ (medium triangles), and $N_k*(\Sigma F_k/\Sigma F_t)$ (light circles).

formula using the proportion of characters, medium triangles for the formula using the proportion of words,

increase recall) and more incorrect classifications (that decrease precision). Finally, we can note that the best equation for the proportional importance of keywords in titles is consistently the proportion of keywords (N_k / N_t), followed closely by the proportion of characters (C_k / C_t), while the proportion of IDF (ΣF_k / ΣF_t) trails in third position.

It is surprising that the IDF measure gives the worst results of the three, when it worked well in other projects [12]. However, the IDF measure is based on a simple assumption, that a word with low semantic importance is one that is used commonly in most documents of the corpus. In our current system however, the "documents" are article titles, which are by design short, limited to important keywords, and stripped of semantically irrelevant words. These are clearly in contradiction with the assumptions that underlie the IDF measure. We can see this clearly when we compare the statistics of the keywords given in the example in [12] with the same keywords in our system, as we do in Table I. The system in [12] computed its statistics from the entire Wikipedia corpus, including article text, and thus computed reliable statistics; in the example in Table I the rarely-used company name WWE is found much more significant than the common corporate nouns chief, executive, chairman and headquartered. On the other hand, in our system WWE is used in almost as many titles as executive and has a comparable F_w score, which is dwarfed by the F_w score of chairman and headquartered, two common words that are very rarely used in article and light circles for the formula using the proportion of IDF. Three results are also shown for each experiment: the overall precision computed using equation (12) in a dashed line, the overall recall of equation (13) in a dotted line, and the overall F1 of equation (14) in a solid line.

A few observations can be made from Figure 2. The first is that the overall result curves of all three variations have the same shape. This means that the system behaves in a very consistent way regardless of the exact formula used. There is no point where one of the results given one equation shoots off in a wildly different range of values from the other two equations. Moreover, while the exact difference in the results between the three equations varies, there is no point where they switch and one equation goes from giving worse results than another to giving better results. We can also see that the precision decreases and the recall increases as we increase the number of acceptable goal categories. This result was to be expected: increasing the number of categories returned in the results means that each query is classified in more categories, leading to more correct classification (that

TABLE I
COMPARISON OF IDF OF SAMPLE KEYWORDS

Keyword	T_w^*	F_w^*	T_w	F_w
WWE	2,705	7.8	657	9.0
Chief	83,977	5.6	1,695	8.1
Executive	82,976	5.8	867	8.7
Chairman	40,241	7.2	233	10.1
Headquartered	38,749	7.1	10	13.2

*Columns 2 and 3 are taken from [12].

two common words that are very rarely used in article titles.

Finally, we can wonder if the two parts of equation (1) are really necessary, especially since the best equation we found for proportional importance repeats the N_k term. To explore that question, we ran the same test again using each part of the equation separately. Figure 3 plots the

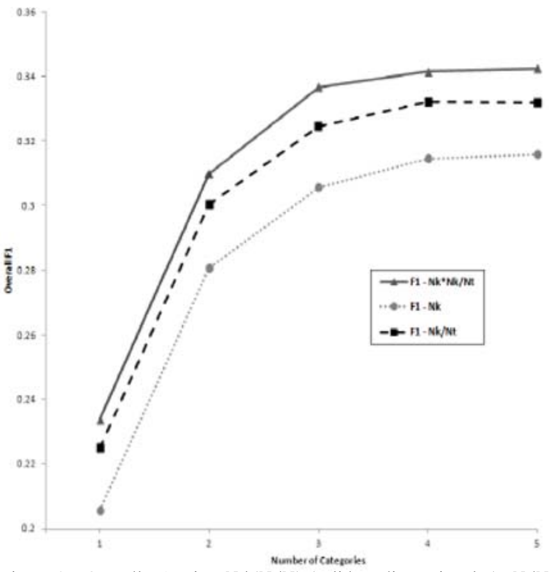

Figure 3. Overall F1 using N_k*(N_k/N_t) (solid medium triangles), N_k/N_t (dashed dark rectangles), and N_k (light dotted circles).