Better Categorization:
Adapted Nearest Neighbor

Författare:
Jakob Svemar, D07, (dt07js6@student.lth.se)
Fredrik Gullstrand, D07, (dt07fg3@student.lth.se
Kursansvarig för EDAN60: Pierre Nugues

Inlämningsdatum: 13-01-2014

Abstract

This paper acts as a report for a project in the course EDAN60 Language Technology: Project. The main goal of the project have been to explore classification of small texts using a variation of the algorithm k-nn. This field is very relevant today with growing needs for programs to be able to recognize and classify sentences without human help. The main target for this program is to classify events on a website called eventful, but it works just as well in any environment where you have a set of categories and the possibility to describe each item. For instance a user based web shop like ebay or posting on a forum. To measure the results a comparison is made with a leading tool called LibShortText. While the results of this experiment were not better than LibShortText, it did reveal some interesting points. Mainly that different categories may benefit from different classification techniques. This was further confirmed when combining both this algorithm and the one LibShortText uses, resulting in an overall better result.

Innehåll

1 Background

With the emergence of the communication culture in the 21st century there has become a larger need to sort information into the correct category. Regardless of who is adding it, a machine or a person. In the same way there is a need to be able to sort existing information for the reciver. All this has to be done without forcing a person to interact with it at every step. Having a program that can categorize texts is the first step.

1.1 Aim

With the thought that categorisation could be a very powerful and needed product we set out to create a program that could categorise a text regardless of origin to set categories. While the program is tailored for the eventful website, any database which offers similar structure for descriptions and categorization can utilize it with small adaptations.

2 Other programs related to the project

This projects goals are very close to what the LibShortText tool does[3]. And as such it formed a good basis of comparison. As previously revealed, LibShortText is also used to enhance the results of our algorithm, as a combination of the two performs better than both tools independently.

2.1 LibShortText

LibShortText is an open source text classification tool developed mainly at the National University of Taiwan[2]. It is light weight and yields good results on the test sets used in this project. Input files in the two tools are virtually the same, making the combining of the two quite easy.

3 Theory

One common vector comparison algorithm is called k-nn, this is the basis for the algorithm in this project. In order to compare different vectors, cosine similarity is used.

3.1 K-nearest neighbors

This algorithm compares vectors using some sort of similarity function[1]. It finds the k closest vectors to the one of interest and checks their category, finally it picks whatever category is in majority.

3.2 Adapted nearest neighbors

One particular flaw in k-nn is that the number of calculations increases quickly when the training set gets bigger. With a training set of 10000 vectors for each category and 10 categories, a total of 100000 separate calculations have to be made for each vector classification. What this algorithm does is combines the

training set vectors into one large. All saved in a hashmap. With the same example as before, to classify one vector will now take 10 separate calculations. Due to how the cosine comparison is designed, the vector size of the model is not important, it is rather the size of the target vector that is interesting. With all vectors

3.3 Method of Comparison: Cosine similarity

The cosine similarity[4] is a quick way for a computer to calculate the angle between two vectors. In figure 1 A and B are vectors and n is the number of dimensions in the vectors.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Figur 1: Formula used for cosine similarity calculations.

4 Implementation

In order to create this project a few things had to be addressed. First of all there was a need for a data set on which to apply the algorithm. This came from a website called eventful.com which is a place where events from all over the world can be found. Secondly an algorithm had to be designed. This project focuses on an adapted version of k-nearest neighbors using cosine similarity. Finally a program had to be created in order to bring these points together.

4.1 Corpus

The corpus is retrieved from the website eventful.com. It was then extracted in chunks one category at a time. during the extraction process the corpus was divided into three parts used in different stages throughout the development.

Training Set The largest part was the training set, comprising of one half of the entire corpus, this was used to train the model in how to recognize and differentiate the different categories.

Developer Test Set The developer set is the set we use to test our algorithm and try to train it into getting as high a score as possible. By doing the main testing on this set and not the final test allowed for a more objective result on the final test set. This set was also used in creating the confusion matrices used to merge the results of this project and that of LibShortText.

Final Test Set The final test set is the test set that was used to finalize the results. Since this set was used no further development occurred in order to preserve the integrity of the results.

4.2 Program description

The whole project is designed in a modular capacity where each program works without the help from the others. This allows a user to switch out any of the modules to one that works better for them. Most of the programs do however demand some form of input which needs to be accounted for.

Extractor The extractor collects the different categories available from eventful and proceeds to extract a set amount of events.

CategoryTrainer Creates the language model from which classifications can be made. It requires a training set in order to work.

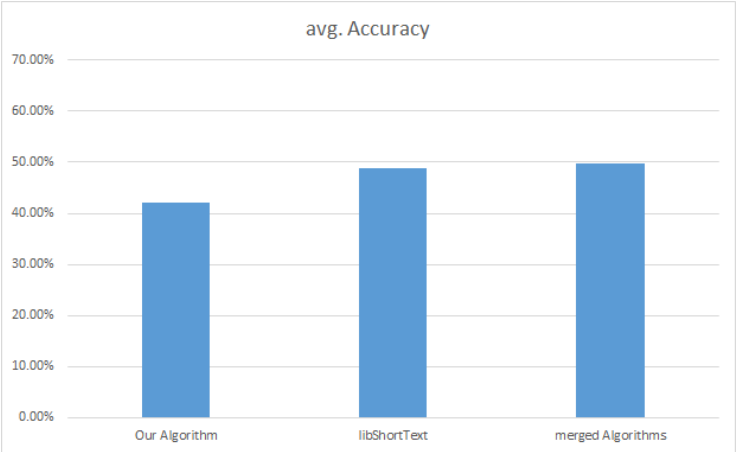
EventCategorizer Uses our algorithm to classify test sets of events. It can produce results in the form of an overall accuracy, precision and recall per category and an confusion matrix showing the probability each type of event have of being classified correctly. It requires a language model in order to work.

EventCategorizerWithLib Virtually the same program as EventCategorizer, with the addition of comparing the confusion matrix from our algorithm and that of LibTextShort. It then continues to pick the answer with the highest probability of being correct. This program needs confusion matrices from both EC and LST run on an development test set before running on a final test set.

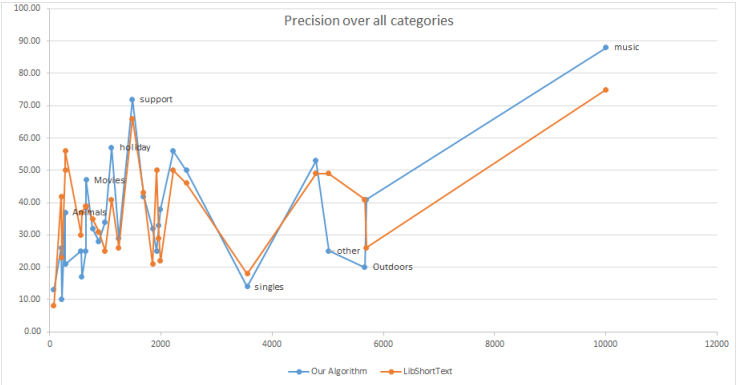
LibShortTextConverter A simple program that converts our data to a format that is acceptable for LibShortText.

5 Results

The results are presented in three parts. Figure 3 shows the precision of each category when using our method compared to LibShortText. On the horizontal axis one can see how many events existed for each category. figure 2 shows the average accuracy of our algorithm, LibShortText and the merged algorithm. In the appendix there is a list showing the precision and recall for our algorithm for each category. Delivered with the document as an attachment are two confusion matrices, one for our algorithm and one for LibShortText for a more detailed view of the results.



Figur 2: Displaying the difference in accuracy between our algorithm, LibShortText and the merged results.



Figur 3: Displaying the different categories and their precision in correlation with the amount training data available for each category.

5.1 Evaluation of results

While observing the results presented in diagram 1 one can see that the results improve when more events exists for the category. Some categories appear are easier to categorise, for example animals (which are very high for its few examples). The is probably because of the high number of keywords such as dog, cat, kennel etc.

While observing diagram 2 one can see how LibShortText is better in most cases compared to our algorithm. Together with diagram 1 this shows that different algorithms have different qualities depending on the category.

6 Conclusion

Firstly, we have concluded that to have any precision in the categorisation one has to have at the very least ten thousand examples of that category. This

becomes clearly evident when looking at figure 3. It also shows that some categories are more easily recognized. Secondly, the results of our cosine similarity, summed KNN algorithm as shown in figure 2 gave a worse overall result than libShortText. The reason why our final results ended up being better than libShortText in the end is because of using libShortText in most cases and our algorithm in those cases where it would give a better result. What we have proved is that different algorithms have varying success depending on the input. Something that should not be too strange. Thirdly, we can conclude that while summing the vectors before calculating KNN made the algorithm faster, it also made it a little less precise.

7 Future work and improvements

As this program is centered around an experiment with an adaptation of the k-nn algorithm, there is a lot of polishing to be made, refactoring and removing unused code but also adding the possibility to configure the program without changing the code would be a good improvement. Features like normalization and tf-idf are currently disabled and require code changing to be enabled. To expand on the program one could also make it more commercial, or usable in a real world application. As it is now the program is designed to take a big file and categorize it in one big chunk. Creating an interface for other programs to use the classification methods should be the first step but also adding different way of displaying the classification results would be interesting, now it gives the models best guess, by adding the models second and third best guess and letting the end user choose the best one could potentially increase the accuracy Significantly. Finally it would be interesting to see if a different similarity measurement was used, for instance Euclidean distance.

Referenser

[1] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46 (3): 175–185. [Accessed 11 January 2014].

[2] Hsiang-Fu Yu, Department of Computer Science, University of Texas at Austin, Austin, TX 78712 USA, Chia-Hua Ho, Yu-Chin Juan, Chih-Jen Lin, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, "LibShortText: A Library for Short-text Classication and Analysis" `http://www.csie.ntu.edu.tw/~cjlin/papers/libshorttext.pdf` [Accessed 11 January 2014].

[3] LibShortText Documentation `http://www.csie.ntu.edu.tw/~cjlin/libshorttext/doc/` [Accessed 11 January 2014].

[4] Singhal, Amit (2001). Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43. [Accessed 11 January 2014].

Appendix A
Category: performing_arts
Precision: 0.2514705882352941
Recall: 0.20284697508896798
Category: outdoors_recreation
Precision: 0.19722222222222222
Recall: 0.21450151057401812
Category: other
Precision: 0.2525083612040134
Recall: 0.1696629213483146
Category: technology
Precision: 0.10457516339869281
Recall: 0.31683168316831684
Category: comedy
Precision: 0.20638297872340425
Recall: 0.7950819672131147
Category: sports
Precision: 0.4056420233463035
Recall: 0.40057636887608067
Category: business
Precision: 0.3182640144665461
Recall: 0.4656084656084656
Category: science
Precision: 0.12849162011173185
Recall: 0.6388888888888888
Category: movies_film
Precision: 0.46867167919799496
Recall: 0.5899053627760252
Category: sales
Precision: 0.33212341197822143
Recall: 0.2900158478605388
Category: learning_education
Precision: 0.381294964028777
Recall: 0.10685483870967742
Category: conference
Precision: 0.5292164674634794
Recall: 0.33347280334728036
Category: clubs_associations
Precision: 0.24608150470219436
Recall: 0.48909657320872274
Category: music
Precision: 0.8839004861309694
Recall: 0.6182
Category: schools_alumni
Precision: 0.4158950617283951
Recall: 0.6447368421052632
Category: books
Precision: 0.288
Recall: 0.7024390243902439
Category: family_fun_kids

Precision: 0.31869918699186994
Recall: 0.21281216069489686
Category: animals
Precision: 0.3701923076923077
Recall: 0.5703703703703704
Category: support
Precision: 0.7176938369781312
Recall: 0.5113314447592068
Category: religion_spirituality
Precision: 0.2832980972515856
Recall: 0.6049661399548533
Category: art
Precision: 0.24519230769230768
Recall: 0.3805970149253731
Category: attractions
Precision: 0.16962025316455695
Recall: 0.475177304964539
Category: community
Precision: 0.5038759689922481
Recall: 0.11403508771929824
Category: singles_social
Precision: 0.1362776025236593
Recall: 0.31718061674008813
Category: holiday
Precision: 0.5738758029978587
Recall: 0.5075757575757576
Category: fundraisers
Precision: 0.5627240143369175
Recall: 0.48307692307692307
Category: politics_activism
Precision: 0.25925925925925924
Recall: 0.5632183908045977
Category: festivals_parades
Precision: 0.336272040302267
Recall: 0.5393939393939394