|  |
| --- |
| **Automatic Coding of HCI Papers using NLP Topic Modeling Techniques** |
|  |
| **Obianuju Okafor**  **Computer Science and Engineering Department**  **University of North Texas**  **obianujuokafor@my.unt.edu** |
|  |
|  |
|  |
|  |

Abstract

|  |
| --- |
| 000  001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049 |

The Human Computer Interaction research field is quite broad, there is a plethora of papers that have been published in this field. When conducting a literature review the process of reading and analyzing such publications and coding them to come up with overarching themes can be quite daunting. As a solution, I propose an automatic method to eliciting the key themes in research papers by leveraging natural language processing techniques. In this paper I will use three NLP techniques LDA, LSA & NMF to auto-code 50 research papers. I would also compare the topics generated with the ones derived from performing the same task manually. The results suggest that NLP techniques can be used to support researchers coding qualitative data. I found that NMF produced the most coherent topics, it was also the most accurate in classifying papers under a topic. I also discovered that NMF and LSA often produced the same results was also the most accurate when coding papers.

Introduction

There are a lot of publications being released today in the Human Computer Interaction (HCI) field. To keep up with the growing trends and innovation and to better understand the evolution of research in this fields literature review is done [1]. Literature review is a basis for research in nearly every academic field. The information gained from literature reviews is useful for several reasons such as managing and organizing research proposals [4], predicting the likely impact of research plans and visualizing research themes [5].

The process of writing a literature review involves several activities; researchers usually have to read and analyze multiple research papers, code them and further categorize each paper under an overarching topic. This process requires a lot of effort and can be quite grueling, especially when the dataset is huge [4]. There are software applications that help in analyzing research papers, however, they are mostly assistive, and still dependent on the the researcher for further interpretation. A possible alternative to help reduce the effort and also biases immanent in manual analysis is to use NLP topic modelling techniques.

|  |
| --- |
| 050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099 |

NLP is a subfield of computer science and linguistics that deals with algorithms, methodologies, and tools to analyze natural language text and studies grammatical, syntactic, and semantic structure of text [4]. One of the primary applications of natural language processing is to automatically extract what topics people are discussing from large volumes of text, this is known as topic modelling. Topic modelling is the process ofextracting the most representative topics occurring in a collection of documents and grouping the documents with the same topic [14].A typical usage of topic modeling is, resume summarization, search engine optimization, recommender system optimization, customer support improvement.

There are several approaches used for topic modeling, the most common are, Latent Semantic Analysis (LSA/LSI), Probabilistic Latent Semantic Analysis (pLSA), Latent Dirichlet Allocation (LDA) and Non-Negative Matrix (NMF). With LDA being the most popular. These approaches also have several variations e.g PLSA, GLDA. In a later section, I will review some of these approaches [13].

With that being said the goal of this paper is to apply NLP topic modelling techniques to automate the coding process usually done during literature review. I will be using three topic modelling techniques LSA, LDA and NMF. I will compare the codes and topics generated by these techniques to the ones generated manually.

|  |
| --- |
| 100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149 |

The rest of the paper is as follows. I introduce the problem in section one. In the next section I discuss the work that has been done in this area. In section 3, I will present my dataset. I give a review of the models I will be applying in this experiment in section 4. In the 5th section, I describe my methodology. I present my results in section 6 and I discuss the limitations of my experiment in section 7. In the 8th section I conclude and talk about my future works.

Related Works

There have been several works relating to how NLP has been used in the qualitative analysis and classification of textual data.

The authors in [1], presented an automatic method for extracting and examining key research themes using natural language processing. They used this method to analyze 8,845 academic papers published over the past 2 decades. They identified and visualized key research themes and used them to highlight trends.

In the paper [3], they perform theme and sentiment analysis of blog entries using statistical text-mining techniques. Theme identification was performed using a novel technique called ‘Best Separators Algorithm’ and sentiment analysis was performed using logistic regression combined with dimension reduction technique (singular value decomposition). Their results showed significant improvement in accuracy compared to popular approaches.

Guetterman et al. [4] conducted a study to compare the benefits/advantages of a traditional qualitative text analysis using qualitative software, an NLP based analysis, and an augmented approach that combines the two methods, in theme analysis. These three methods were applied to data generated from message survey questions. They found that NLP methods were able to validate major themes found with traditional qualitative analysis but were not useful in identifying nuances. However traditional qualitative text analysis added important details and context.

The authors in [5] applied latent semantic analysis (LSA) in the analysis of transcripts of interviews in order to automate the coding of interview transcripts into higher level concepts to be investigated. In the paper they present two LSA-based algorithms. They evaluate the algorithms against two separate real-life cases taken from the automobile industry and from the Austrian mobile phone market. They compared their results against marketing expert judgements and show that the algorithms proposed provide perfect reliability with appropriate validity in automated coding and textual analysis.

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

The paper [6] presents a hybrid approach to classification of natural language textual data that combines automated data extraction with human qualitative analysis in the. In the framework, they iteratively develop a set of categories to classify the text messages in a mailing list. Next, they use inter-rater agreement measures to refine the list of categories until a high degree of agreement is achieved. The refined categories are then used to classify a representative sample of the textual data. They claim their approach is transparent, generalizable, and suitable for large projects.

The authors in [7] propose a semi-automatic tool that initially uses natural language processing (NLP) and machine learning (ML) techniques for automatic coding, then follows it up with which human analysis for validation and correction. They discuss design strategies adopted to optimize the system performance in automatic coding.

In [8], the authors demonstrated how to use NLP and ML to assist the qualitative researcher in improving the reproducibility and traceability of the coding process through recommendations. They integrated an already existing commercial machine–learning (ML) based concept extraction service into an NLP pipeline. They applied their prototype in three qualitative studies to evaluate its ability to support researchers by giving recommendations congruous with their initial work. They claim that their approach can be applied to interviews from any domain, without the need for researcher to create new rulesets.

Crowston and his team were one of the first to demonstrate the use of high-level NLP techniques for qualitative data analysis. In [9], they presented a case study in which they built a rule–system to perform auto-coding for 12 predefined codes on a set of instant messaging logs and compared the results to a manual coding. Similarly, in [11] they present a case study of the use of NLP for qualitative analysis in which the NLP rules showed good performance on a number of codes.

In a related work [10], they explore semi-automatic coding of textual data by leveraging (NLP) techniques using the same data in [9] [11]. They compare the performance of human-developed NLP rules to those inferred by machine learning (ML) algorithms. The results suggest that NLP with ML may be useful to support researchers coding qualitative data. They concluded that both approaches show promise but they also have shortcomings.

|  |
| --- |
| 200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249 |

Dataset

The dataset used in this paper consisted of 50 HCI related papers published in the last two decades. The oldest paper was published in 1995. The selected papers were relating to providing accessibility for the visually impaired people in computing. These papers were gotten from online databases. They were either published at a conference or in a journal.

Models

This paper focuses on 3 widely used topic modeling approaches:

1. Latent Dirichlet Allocation
2. Latent Semantic Analysis
3. Non-negative Matrix Factorization

Each of these models were designed with different goals and are supported by different statistical theories. However, all 3 topic models are based on the same basic assumption:

* each document consists of a mixture of topics, and
* each topic consists of a collection of words.

LDA

LDA or Latent Derelicht is a generative probabilistic model that assumes that documents are probability distributions over latent topics, and topics are probability distributions over words [13].

Based on these assumptions, we can describe the generative process of LDA as, given the M number of documents and N number of words, represented as a Document-Term Matrix A, and prior T number of topics, the model trains to output:

* ϕ, the word distribution for each topic t
* θ, the topic distribution for each document *i.*

|  |
| --- |
| 250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299 |

* , the topic for each , the *j*-th word in document *i.*

{\displaystyle w\_{ij}}These two distributions are based on two hyper parameters. θ is based on Alpha parameter *α,* which is Dirichlet prior concentration parameter that represents document-topic density. ϕ is based on **Beta parameter** *β*, which is the prior concentration parameter that represents topic-word density.

## **LSA**

Latent semantic Analysis or  Latent semantic indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text [21]. LSI is based on the assumption that words that are used in the same contexts tend to have similar meanings.

Similar to LDA, given M documents and N words in our vocabulary, we can construct an m × n document-term matrix *A*, in which each row represents a document and each column represents a word. Unlike LDA, LSA models typically replace raw counts in the document-term matrix with a term frequency-inverse document frequency (tf-idf) score. Tf-idf assigns a weight for term *j*in document *i*as follows:

where is the number of occurrences of a term in the document, N is the total number of documents and is the number of documents containing term.

A= US

Using SVD, the matrix A is decomposed into three partial matrices U, S, and VT(transpose of matrix V).

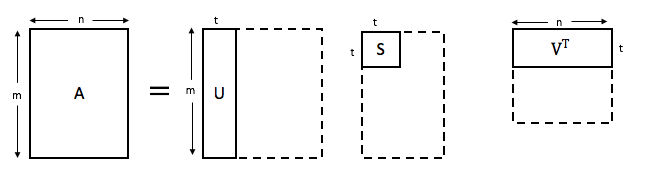


Fig 2: LSA Matrix Decomposition

|  |
| --- |
| 300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349 |

where U is our document-topic matrix, S is a diagonal matrix of the singular values of A, and V is the term-topic matrix. In both U and V, the columns correspond to each of the *k* topics. In U*,*rows represent document vectors expressed in terms of topics; in V, rows represent term vectors expressed in terms of topics [22]. The length of each vector would be k. These vectors are used to find similar words and documents using the cosine similarity method [16].

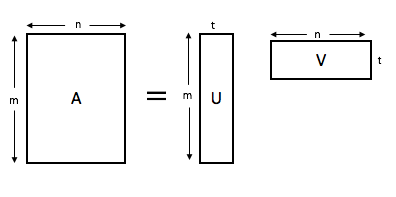
## **NMF**

As the name implies, non-negative matrix factorization is a linear algebraic method where a non-negative matrix is factorized into two non-negative matrices.

Like the other two models, NMF takes as input the m x n term-document matrix A. Each element in the document represented the weight of a certain word. The weight might be the raw count like in LDA or the tf-idf weighted count like in LSA.

A ≈ U V

NMF factorizes the matrix A to yield two non-negative factors U and V. Where, the U factor contains document weights across the t topics. Each row corresponds to a different document, and each column corresponds to a topic. The V factor contains term weights relative to t topics. Each row corresponds to a topic, and each column corresponds to a unique term in the corpus. The V factor contains term weights relative to each of the t topics. Each row corresponds to a topic, and each column corresponds to a unique term in the corpus vocabulary.



|  |
| --- |
| 350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399 |

Fig 3: NMF Matrix Factorization

By default, the values in factors U and V are given random initial values and then updated according to the following iterative update rules:

U = U V = V

|  |
| --- |
|  |
| Figure 1: LDA Model representation**.** |

# **Methodology**

In this paper, I compare the three NLP topic modeling techniques. I compare the results they generated to a gold standard, the codes I generated manually. I give further details of my approach in the next subsections.

## **Document Collection**

The first phase of this project involved gathering a selection of papers for analysis. This was done me and another PhD student in my laboratory. The objective was to collect the full text of HCI papers relating to the concept of accessibility. The ACM, IEE and Google scholar database was used to identify academic papers with the phrases *“Accessibility”,* *“Computer science”,* *“blind programmers”,* “*block-based programmers*”, “accessible”, “*programming environments”,* in their title, abstract or keywords. If any of the key words were found, we downloaded the full text of the publication freely using the university’s access.

After gathering the papers, we quickly schemed through them to see if they were relevant or not. At the end of this process we were able to come up with a total of 50 papers. The contents of each paper was entered into an excel sheet with columns containing the title, authors name, year of publication and conference, abstract and conclusion if exists. This excel file was then converted to a CSV data file to enable it to be parsed by my methods.

## **Human Annotation**

|  |
| --- |
| 400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449 |

Having collected the papers, the next step was to manually assign codes. Before starting, we created an initial set of codes with clear definitions which we expected to come across. We then performed the coding task independently. While reading each paper, we assigned codes that were most representative of the contents of the paper. The list of codes grew as we read more papers and discovered more concepts [6].

After we were both done, we compared the codes assigned to each paper. On papers where we were in disagreement, we jointly revaluated the paper and came to an agreed on the codes to be assigned. This process is repeated until an acceptable level of agreement was achieved on all papers. The goal was to achieve a high degree of agreement among ourselves on which set of code to be assigned to a paper.

The last step in this process was to iteratively coalesce codes with similar meaning into a single category, and assign a new code to the category. When the list has coalesced into a handful of categories with distinct meanings, we then assigned papers to each category. Some papers fell under multiple categories and therefore had multiple codes.

## **NLP-based Annotation**

In this phase we perform the coding of our papers using NLP topic modelling algorithms. First, we preprocess our data. Then we apply our algorithm on the pre-processed data. I also perform parameter tuning on the number of topics.

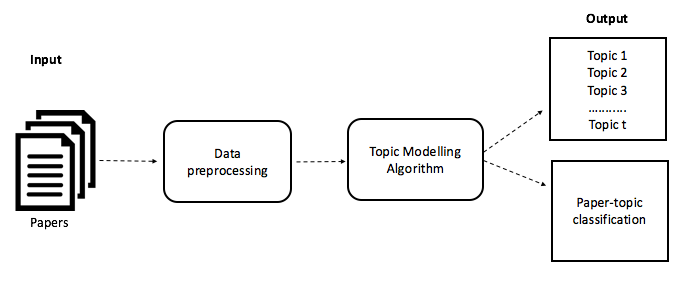


Fig 4: Methodology

* + 1. Data Cleaning

First the csv data file is loaded on Jupyter notebook. Before beginning preprocessing, I cleaned up the data a bit. (show table). Since the goal of this analysis is to perform topic modeling, I dropped the unnecessary columns like ‘ID’, ‘Reference’, ‘Author’, ‘Year’, ‘Conference/ Journal’, and focused solely on the **‘**Abstract’ and ‘Conclusion**’** columns of each paper entry. For papers with no conclusions, I filled in the empty cell with the text “No conclusion”. Next, I merged the two columns abstract and conclusion to form a new column called ‘Paper\_text’. The text in this new column is what I would apply the topic models to.

|  |
| --- |
| 450  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499 |

* + 1. Preprocessing

After cleaning up the data, I applied a couple of simple preprocessing techniques on the content of ‘Paper\_text’ column to make it more amenable for analysis, and improve the reliability of results. First, I tokenized each sentence into a list of words. Next, I used a regular expression to remove any punctuation present in the tokens. I then converted all tokens to lowercase. Additionally, I removed the tokens of length < 3, and tokens that were stop words i.e. words that are frequent and do not convey useful information. Each token was then lemmatized i.e. reduced to its most basic form. Based on the tokens that frequently occurred together, I used Gensim’s Phrases model to create Bigrams and Trigrams, examples include “computer\_science”, “visually\_impaired”, “Programming\_language”. Lastly, I combined the tokens back into sentences.

* + 1. Exploratory analysis

To better understand the data and to confirm that the preprocessing was done correctly, I performed an exploratory analysis. I created a word cloud using the wordcloud package to get a visual representation of most common words in the corpus. This can be seen in figure 1, the most significant words are depicted the largest and smaller words less prominent. In fig 6. I also used a barchart to display the top 10 most common words. The most frequent word with a count of 175 is the word ‘blind’.

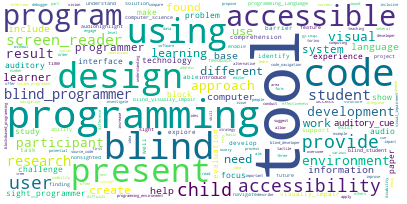
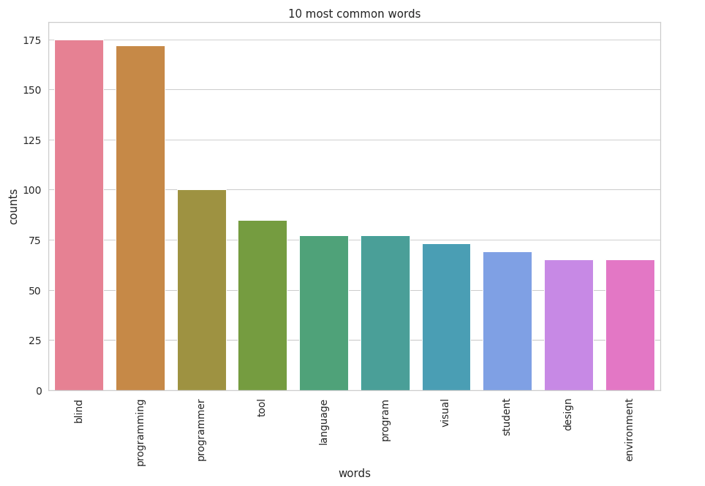


Fig 5: Word cloud



|  |
| --- |
| 500  501  502  503  504  505  506  507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549 |

Fig 6: Barchart of top 10 most frequent words

* + 1. Topic Modelling Algorithm

|  |
| --- |
| 550  551  552  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596  597  598  599 |

In this step, I applied the three topic modelling techniques LDA, LSA, &NMF to my data. These three models take as input a document-term matrix (Bag of Words BOW). Using sklearn's TfidfVectorizer for my NMF and LSA model, and Vectorizer for my LDA model, I transformed my data to a document-term matrix of 1000 terms. I could have used all the terms in my corpus to create this matrix but that would require a lot of computation. Hence, I restricted the number of features to 1000. The difference between TfidfVectorizer and Vectorizer, is that TfidfVectorizer replaces raw counts of each term in the document-term matrix with a term frequency-inverse document frequency (tf-idf) score.

The next step in each model is to decompose the corresponding document-term matrix into multiple matrices. In LDA and NMF I use sklearn’s decomposition models LatentDirichletAllocation and NMF respectively. For LSA, I use sklearn's TruncatedSVD for matrix decomposition. The number of topics in each model can be specified by using the *n\_components* parameter. In this experiment I selected 10 as my number of topics. I talk about why I selected this number in the next subsection.

The resulting matrices derived after running each topic model are the document-topic matrix and term-topic matrix. In the term-topic matrix, sorting the rows in reverse, reveals the top terms for each topic. The document-topic matrix represents the “weights” of each topic in the document, using these weights we are able to classify a document under the topic with the most weight.

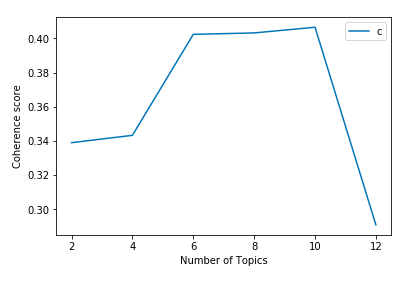
5.3.5 Parameter Tuning

The parameter selection decision for topic modelling involves choosing the optimum number of topics t. The most common approach involves measuring and comparing the topic coherence of the models generated for different values of t.

Topic Coherence measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are outcomes of statistical inference [18].

|  |
| --- |
| 600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616  617  618  619  620  621  622  623  624  625  626  627  628  629  630  631  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646  647  648  649 |

There are various metrics for measuring topic coherence e.g. NPMI, UMass, UCI, C\_V. To measure the topic coherence of my model, I used the C\_V metric from gensim’s built-in library. I selected the LSI model for this task and used the values of k from 2 to 12. I found that the t with the highest coherence was t= 10. Therefore, I selected 10 as my number of topics. The second best value for k was 8.



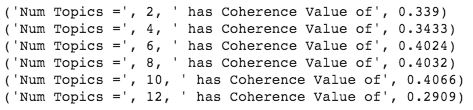


Fig 7. Coherence scores for different number of topics

# **RESULTS**

In this section I discuss the results generated after running each of the models on my dataset. I break down the results into 3 sections, topic similarity, topic coherence, and model accuracy. Table 1, 2, and 3 shows the 10 topics along with their top 10 terms. Based on the top words for each topic we can infer what the topic is about, for example topic 5 of the LDA Model is about the challenges blind student face in computer science.

## **Topic Similarity**

|  |
| --- |
| 650  651  652  653  654  655  656  657  658  659  660  661  662  663  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690  691  692  693  694  695  696  697  698  699 |

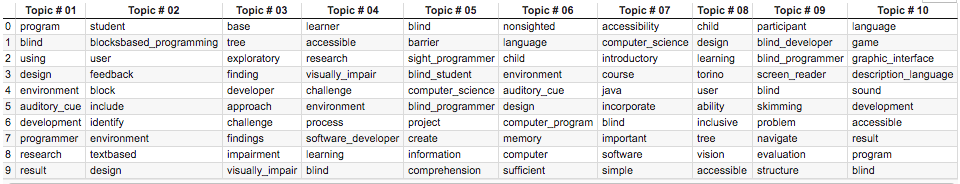
## Looking at the table 1, 2, and 3 above, we can observe that there are recurring topics across the 3 models, for example topic #02 in LDA model, Topic #05 in the LSA model and topic #05 in the NMF models are talking about the same concept which is accessible block-based programming environment. Likewise, topic #05 in LDA, topic #06 in LSA and topic #07 in NMF are all on the barriers blind students face in computer science. Similarly, topic #10 in LDA and LSA corresponds to topic #08 in the NMf model.

Additionally, from table 4. We can see that LSA and NMF not only produced similar topics, but also topics with the same top words. i.e topic #08 in LSA and topic #09 in NMF, topic #06 in LSA and topic #07. Topic #04 in both LSA and NMF. Looking at the top words in these matching topics, we can see that over 50% of the top words are the same. This can be seen across several topics in both of these models.

## **Topic Coherence**

After evaluating each of the topics produced by the 3 models, I found that NMF sometimes produces more meaningful topics out of the 3. Looking at the three tables above. We can see that there is some coherence between the words in each topic produced by the NMF model. For example, Topic #02 in NMF shows words associated with creating an inclusive physical environment for learning, Topic #04 shows words associated with an audio debugger for computer programs. NMF also produced highly distinguishable topics, each topic it produced were unique and involved a new concept.

On the other hand, comparing the results of LDA to NMF. We can see that NMF performs better. The topics created by the LDA was not very coherent. For some topics looking at the top words it is hard to say what that topic is about, for example Topic #07. The LDA model also produced topics with repeated words, for example topic #03 and Topic #04 we can see there are repeated words like challenges, developer, visually impair. These two topics describe the same concepts. Likewise, topic #01 and topic #06.



|  |
| --- |
| 750 751  752  753  754 755  756  757  758  759  760  761  762  763 764  765  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799 |

|  |
| --- |
| 700 701  702  703  704 705  706  707  708  709  710  711  712  713 714  715  716  717  718  719  720  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736  737  738  739  740  741  742  743  744  745  746  747  748  749 |

Table 1: LDA Model topics and top 10 words

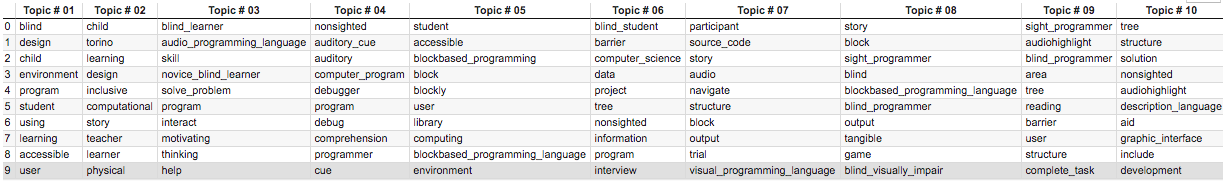


Table 2: LSA Model topics and top 10 words

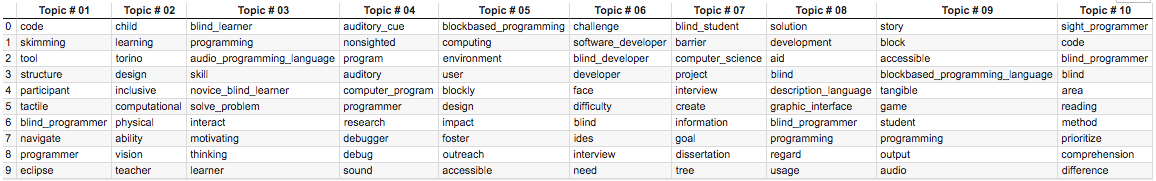


Table 3: NMF Model topics and top 10 words

LSA performed almost as good as NMF. Looking at the topics it produced, it is easy to tell what the topic is about, for example topic #06 is on the challenges blind developers face in accessible block-based programming environments. Looking at the top words in each topic they all have semantic similarity.

Along with producing coherent topics, LSA also produced highly distinguishable topics with the exception of topic #01, all other topics catered to different concepts. However, LSA did repeat some key words across topics like the word “audiohighlight” in topic #09 and topic #10. Having read the papers, I know the word “audiohighlight” belongs in topic #09 and not topic #10.

## **Model Accuracy**

In last part of this experiment I evaluated the models for accuracy. I compared the results produced by each of the 3 models to the gold standard which is my manually annotated data. Before I applied the topic modelling techniques to my data, I manually assigned codes to each paper, these codes are the most representative topics in the paper. These codes are the basis of comparison when evaluating the topic models for accuracy.

|  |
| --- |
| 850 851  852  853  854 855  856  857  858  859  860  861  862  863 864  865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899 |

|  |
| --- |
| 800 801  802  803  804 805  806  807  808  809  810  811  812  813 814  815  816  817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LSA Topic #04 | NMF Topic #04 | LSA Topic #06 | NMF Topic #07 | LSA Topic #05 | NMF Topic #05 |
| nonsighted | auditory\_cue | blind\_student | blind\_student | student | blockbased\_programming |
| auditory\_cue | nonsighted | barrier | barrier | accessible | computing |
| auditory | program | computer\_science | computer\_science | blockbased\_programming | environment |
| computer\_program | auditory | data | project | block | user |
| debugger | computer\_program | project | interview | blockly | blockly |
| program | programmer | tree | create | user | design |
| debug | research | nonsighted | information | library | impact |
| comprehension | debugger | information | goal | computing | foster |
| programmer | debug | program | dissertation | blockbased\_programming\_language | outreach |
| cue | sound | interview | tree | environment | accessible |
| 8 out of 10 | | 7 out of 10 | | 6 out of 10 | |

Table 4: Topic similarity between LSA and NMF

Out of all the 3 models, NMF was the most accurate in classifying papers under the right topic, while LSA performed the worst when classifying papers under the right topics. NMF was able to achieve an accuracy of …..

[I AM STILL WRITING THIS SECTION]

# **LIMITATIONS**

My research study had the following limitations; Firstly, the experiment in this study was conducted using only the abstract and conclusion of each of the selected research papers, as opposed to using the complete text of the paper. Also, the dataset in this study consisted of only 50 research papers, this is considerably small compared to the dataset in other studies. Additionally, the 50 selected papers were closely related, they were all HCI papers relating to the concept of accessibility.

Notwithstanding, the models performed quite well, particularly, the NMF model. In a previous study it has been found that NMF sometimes produces more meaningful topics for smaller datasets. This could infer that parsing the entire text of a paper and a large dataset is not required for good performance of topic models.

# **CONCLUSIONS AND FUTURE WORK**

NLP techniques have been found to be helpful in the analysis and coding process done during literature reviews. They provide objective and require less effort as they are mostly automated.

This research project investigated the automatic coding of 50 HCI research papers published in the past 2 decades using NLP topic modelling techniques. The 3 topic modelling techniques used were LSA, LDA & NMF. The outcomes of these modelling techniques were evaluated and compared not only to one another but also to a golden standard which was manually annotated.

The results prove that NLP techniques are truly helpful to researchers when coding qualitative data. In particular, NMF was found to produce the most comprehensible topics, it also had the highest accuracy in paper topic classification. Another finding was that the topic models often produced the similar results, especially the models LSA and NMF.

A future work would be to apply the topic modelling algorithm on not only the abstract and conclusion of a paper but on the entire paper text. A comparison would then be made to see if more data improves the accuracy of the models. Additionally, I also plan to implement a supervised version of the topic models, where the topic models are fed training data with topics as labels, and asked to predict the labels (topics) of the test data.

References

[1] Kenneth Cosh, Sakgasit Ramingwong, Narissara Eiamkanitchat, Lachana Ramingwong. *Automatically Identifying Themes and Trends in Software Engineering Research*, IEEE, 2018.

|  |
| --- |
| 900 901  902  903  904 905  906  907  908  909  910  911  912  913 914  915  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945  946  947  948  949 |

[2] Patrick J. Tierney. *A Qualitative Analysis Framework Using Natural Language Processing and Graph Theory* 2012.

[3] Jayanta Kumar Pal, Abhisek Saha. Identifying Themes in Social Media and Detecting Sentiments. International Journal of Statistics and Applications: 2011; 1(1): 14-19 DOI: 10.5923/j.statistics.20110101.04.

[4] Timothy C Guetterman, PhD; Tammy Chang, MD, MPH, MS; Melissa DeJonckheere, PhD; Tanmay Basu, PhD; Elizabeth Scruggs, MD; VG Vinod Vydiswaran, PhD. Augmenting Qualitative Text Analysis with Natural Language Processing: Methodological Study.

[5] Ingo Feinerer. Automated Coding of Qualitative Interviews with Latent Semantic Analysis. 2006.

[6] John Noll, Dominik Seichter, Sarah Beecham. A Qualitative Method for Mining Open Source Software Repositories. 8th International Conference on Open Source Systems (OSS), Sep 2012, Hammamet, Tunisia. pp.256-261, 10.1007/978-3-642-33442-9\_18 . hal-01519053

[7] Yan, J. L. S., McCracken, N., & Crowston, K. (2014). Semi-Automatic Content Analysis of Qualitative Data. In *iConference 2014 Proceedings* (p. 1128–1132). doi:10.9776/14399

[8] Andreas Kaufmann, Ann Barcomb, Dirk Riehle. Supporting Interview Analysis with Autocoding

[9] K. Crowston, E. E. Allen, and R. Heckman, “Using natural language processing technology for qualitative data analysis,” *International Journal of Social Research Methodology*, vol. 15, no. 6, pp. 523–543, 2012.

[10]  K.Crowston,X.Liu,andE.E.Allen,“Machinelearning and rule-based automated coding of qualitative data,” *Proceedings of the American Society for Information Science and Technology*, vol. 47, no. 1, pp. 1–2, 2010.

[11] Kevin Crowstona\*, Eileen E. Allena, Na, “Lina” Lib, Michael J. Scialdonea and Robert Heckmana. Using natural language processing technology for qualitative data analysis. 2010.

[12] Liddy, E. D. (2003). Natural Language Processing *Encyclopedia of Library and Information Science* (2nd ed.). New York: Marcel Decker.

[13] <https://stackabuse.com/python-for-nlp-topic-modeling/>

[14] <https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0>

|  |
| --- |
| 950 951  952  953  954 955  956  957  958  959  960  961  962  963 964  965  966  967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990  991  992  993  994  995  996  997  998  999 |

[15] <https://medium.com/mlreview/topic-modeling-with-scikit-learn-e80d33668730#.vivglhmhv>

[16]<https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/>

[17]<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>

[18] Keith Stevens Philip Kegelmeyer David Andrzejewski David Buttler. Exploring Topic Coherence over many models and many topics. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 952–961, Jeju Island, Korea, 12–14 July 2012 <https://www.aclweb.org/anthology/D12-1087.pdf>

[19]<https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python>

[20] <https://stackabuse.com/python-for-nlp-working-with-the-gensim-library-part-2/>

[21] Wikipedia contributors. (2020, February 7). Latent semantic analysis. In *Wikipedia, The Free Encyclopedia*. Retrieved 12:38, April 6, 2020, from <https://en.wikipedia.org/w/index.php?title=Latent_semantic_analysis&oldid=939659515>

[22]<https://medium.com/@souravboss.bose/comprehensive-topic-modelling-with-nmf-lsa-plsa-lda-lda2vec-part-1-20002a8e03ae>

[23] <http://derekgreene.com/slides/topic-modelling-with-scikitlearn.pdf>