Rejects

Natural language,” as the terms suggests, is language spoken or written by humans, as opposed to a language used to program or communicate with computers. Natural language processing (NLP) falls under the rubric of arti cial intelligence (AI), which is the sub eld of computer science concerned with the concepts and methods of symbolic inference by com- puter and symbolic knowledge representation for use in making inferences. Natural lan- guage understanding by computers is one of the hardest problems of arti cial intelligence due to the complexity, irregularity and diversity of human language, and the philosophical problems of meaning (natural language, n.d.). AI can be seen as an attempt to model as- pects of human thought on computers (arti cial intelligence, n.d.). (This report does not attempt to explain the very complex computational theories, processes, or algorithms that underpin NLP software. Instead, NLP applications are treated as a black box, with a brief description of how a qualitative researcher would use NLP software.) [2]

Natural Language Processing is a computational approach to text analysis. It “is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications” (Liddy, 2003). [9] [12]

One of the primary applications of natural language processing is to automatically extract what topics people are discussing from large volumes of text.

Topic modelling is **Tagging**, abstract “topics” that occur in a collection of documents that best represents the information in them. There are several existing algorithms you can use to perform the topic modeling. The most common of it are, *Latent Semantic Analysis (LSA/LSI), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA)* [14]

Topic modeling is an unsupervised technique that intends to analyze large volumes of text data by clustering the documents into groups. A typical example of topic modeling is clustering research papers that belong to the same topic, just like we would be doing in this paper. In other words, cluster documents that have the same topic. In the case of topic modeling, the text data do not have any labels attached to it. Rather, topic modeling tries to group the documents into clusters based on similar characteristics. [13]

Discovering topics are beneficial for various purposes such as for clustering documents, organizing online available content for information retrieval and recommendations. Multiple content providers and news agencies are using topic models for recommending articles to readers. Similarly recruiting firms are using in extracting job descriptions and mapping them with candidate skill set. If you see the data scientist job, which is all about extracting the ‘knowledge’ from a large amount of collected data. Usually, collected data is unstructured. You need powerful tools and techniques to analyze and understand a large amount of unstructured data. It helps in discovering hidden topics in the document, annotate the documents with these topics, and organize a large amount of unstructured data. [19]

Topic Modeling automatically discover the hidden themes from given documents. It is an unsupervised text analytics algorithm that is used for finding the group of words from the given document. These group of words represents a topic. There is a possibility that, a single document can associate with multiple themes. for example, a group words such as 'patient', 'doctor', 'disease', 'cancer', ad 'health' will represents topic 'healthcare'. Topic Modeling is a different game compared to rule-based text searching that uses regular expressions. [19]

Unlike rule based approaches, our process can be applied to interviews from any domain, without additional burden to the researcher for creating a new ruleset. Our work using three example data sets shows that this approach shows promise for a real–life application, but further research is needed. Our work does not introduce novel ML techniques, but applies established ML-based services like a concept extraction API, to a new domain by integrating them into a NLP pipeline using linguistic and statistical filtering of candidates, in order to improve the researcher’s performance. Our algorithm uses previously coded interviews as training data and extracts the semantic context of each applied code in order to propose codes in new data. The contribution of this work is to provide an initial exploration into the feasibility of using NLP and ML to assist the qualitative researcher in improving the reproducibility and traceability of the coding process through recommendations.

Having collected the papers, the next step was to create a corpus of word frequencies. The corpus represents the expected frequencies for words in papers about software engineering. Clearly the commonest words, like ‘the’ are stop words and don’t differentiate a paper from other papers, so each document can then be compared with the corpus to identify the significant words in the document. In some studies the stop words are simply removed from the text, but some studies have shown that stop words like ‘his’ or ‘her’ can be useful for categorizing documents [15], so here they were left in and a log likelihood comparison discussed next used to diminish their significance. Whilst all the papers are in the field of software engineering, some papers will use words like ‘test’, ‘suite’ and ‘coverage’ more frequently, while others might use words like ‘project’, ‘risk’ and ‘management’. Identifying the significantly overused words in each document creates a profile for each paper.

Word similarity can be computed based on the relative distance between words in a hierarchical word ontology known as WordNet. WordNet is a database for the English language that contains words and multiword phrases and organizes nouns, verbs, adjectives, and adverbs into more than 117,000 synonym sets (known as synsets) A synset is a set of words that have the same meaning (cognitive synonyms). he detailed description of the proposed NLP framework to find different clusters of similar words from the short messages is as follows:

1. message texts and converted to lower case. We did not consider the other parts of speech because they were found to have little contribution toward identifying concepts from the SMS text messages. The same was observed empirically.

2 A vocabulary was created with unique nouns and adjectives (ie, multiple occurrences of a word are discarded).

3 Synsets of each word in the vocabulary are generated using WordNet. Note that a word may have more than one synset as described earlier.

4 A pair of words were grouped together if the Wu-Palmer similarity between any pair of synsets, one generated from the first word and the other from the second word, is >0.9. It may be noted that the Wu-Palmer similarity score ranges from 0 to 1, both inclusive, and 1 indicates the highest similarity. This step was repeated for all pairs of words in the vocabulary.

5 Derivationally related forms of each word in the word pairs were generated, for example, “honest” is derivationally related to “honesty” as generated by WordNet.

6. The most similar pair of words and their derivational forms were combined to create a cluster. This led to several word clusters to be created from the vocabulary.

7 A pair of clusters was merged, if they had at least 50% common members (ie, words). The process continued until no more merges could take place. The method terminated automatically upon satisfying the given condition and generated the final clusters of words. These clusters indicated different senses and semantic meanings present in the given database of SMS text messages [4].

*M* denotes the number of documents

*N* is number of words in a given document (document *i* has {\displaystyle N\_{i}} words)

*K* denotes the number of Topics

*α* is the parameter of the Dirichlet prior on the per-document topic distributions

*β* is the parameter of the Dirichlet prior on the per-topic word distribution

{\displaystyle \theta \_{i}}θ is the topic distribution for document *i*

{\displaystyle \varphi \_{k}}ϕ  is the word distribution for topic *k*

{\displaystyle z\_{ij}} is the topic for the *j*-th word in document *i*

{\displaystyle w\_{ij}} is the specific word.

The LDA is based upon two general assumptions [13]:

* Documents that have similar words usually have the same topic
* Documents that have groups of words frequently occurring together usually have the same topic.

Mathematically, the above two assumptions can be represented as:

* Documents are probability distributions over latent topics. Topics are probability distributions over words. [13]

We can describe the generative process of LDA as, given the *M*number of documents, *N*number of words, and prior *K* number of topics, the model trains to output:

*psi*, the distribution of words for each topic *K*

*phi*, the distribution of topics for each document *i*

The basic idea behind LSA is that texts contain a semantic structure which, however, is obscured by variations in the wording, and that this structure of meaning can be (partially) unveiled by calculating conceptual indices derived through a truncated singular value decomposition. Comparing both texts and terms on the basis of the resulting lower-dimensional space thereby form the basic working principle of analytic applications using LSA [5]

In the simplest version of LSA, each entry can simply be a raw count of the number of times the *j*-th word appeared in the *i*-th document. In practice, however, raw counts do not work particularly well because they do not account for the *significance* of each word in the document.

The various LSA application processes share a common core: typically first a document-term matrix M (also called ‘textmatrix’) is constructed from a given collection of texts denoting the number of occurrences of a specific term (the rows) for each document (the columns). Using a singular value decomposition (SVD), this matrix is decomposed into three partial matrices (the so-called ‘latent-semantic space’). [5]

Latent Semantic Analysis (Landauer and Dutnais, 1997; Landauer et al., 1998) learns topics by first forming a traditional term by document matrix used in information retrieval and then smoothing the counts to enhance the weight of informative words. Based on the original LSA model, we use the LogEntropy transform. LSA then decomposes this smoothed, term by document matrix in order to generalize observed relations between words and documents. [18]

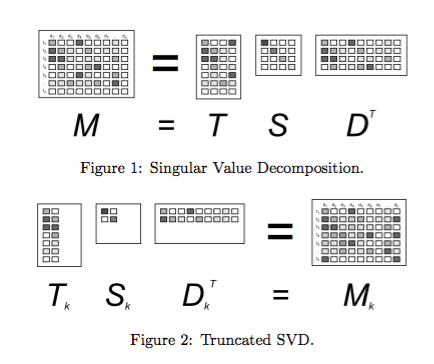
Given the original matrix **A**, we can obtain two matrices **W**and **H**, such that A= WH. NMF has an inherent clustering property, such that W and H represent the following information about A:

A (Document-word matrix) — input that contains which words appear in which documents.

W (Basis vectors) — the topics (clusters) discovered from the documents.

H (Coefficient matrix) — the membership weights for the topics in each document

Thereby T constitutes the left-singular vectors, D the right-singular vectors being both orthonormal and a diagonal matrix S which contains the singular values, so that M = TSDT. By truncating these matrices to a relatively small number of singular values, the document-term matrix M can be restructured in such a way that it reflects only those k common dimensions that account for the greatest share of its underlying variance (cf. (VBRGK06)): Mk = TkSkDkT (see Figure 2). [5]



Each row of the matrix **Uk (document-term matrix)** is the vector representation of the corresponding document. The length of these vectors is k, which is the number of desired topics. Vector representation for the terms in our data can be found in the matrix **Vk (term-topic matrix) [16]**

**NMF is a** linear-algebraic model, that factors high-dimensional vectors into a low-dimensionality representation. Similar to Principal component analysis (PCA), NMF takes advantage of the fact that the vectors are non-negative. By factoring them into the lower-dimensional form, NMF forces the coefficients to also be non-negative.

The NMF decomposition of the term-document matrix would yield components that could be considered “topics”, and decompose each document into a weighted sum of topics. This is called topic modeling and is an important application of NMF.

In order to conduct NMF we formalize an objective function and iteratively optimize it. NMF is an NP-hard problem in general, so we will aim for a good local minima.

https://miro.medium.com/max/446/0*jBGkfjeSZY0DMmu6

A commonly used method of optimization is the multiplicative update method. In this method

Next, let’s work to transform the textual data in a format that will serve as an input for training LDA model. We start by converting the documents into a simple vector representation (Bag of Words BOW). Next, we will convert a list of titles into lists of vectors, all with length equal to the vocabulary.

After running each technique I evaluated them for model perplexity and [topic coherence](https://rare-technologies.com/what-is-topic-coherence/) , this provides a convenient measure to judge how good a given topic model is. In my experience, topic coherence score, in particular, has been more helpful. [17]

What is coherence?

A set of statements or facts is said to be coherent, if they support each other. Thus, a coherent fact set can be interpreted in a context that covers all or most of the facts. An example of a coherent fact set is "the game is a team sport", "the game is played with a ball", "the game demands great physical efforts"

NMF

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | code | child | blind\_learner | auditory\_cue | blockbased\_programming | challenge | blind\_student | solution | story | sight\_programmer |
| 1 | skimming | learning | programming | nonsighted | computing | software\_developer | barrier | development | block | code |
| 2 | tool | torino | audio\_programming\_language | program | environment | blind\_developer | computer\_science | aid | accessible | blind\_programmer |
| 3 | structure | design | skill | auditory | user | developer | project | blind | blockbased\_programming\_language | blind |
| 4 | participant | inclusive | novice\_blind\_learner | computer\_program | blockly | face | interview | description\_language | tangible | area |
| 5 | tactile | computational | solve\_problem | programmer | design | difficulty | create | graphic\_interface | game | reading |
| 6 | blind\_programmer | physical | interact | research | impact | blind | information | blind\_programmer | student | method |
| 7 | navigate | ability | motivating | debugger | foster | ides | goal | programming | programming | prioritize |
| 8 | programmer | vision | thinking | debug | outreach | interview | dissertation | regard | output | comprehension |
| 9 | eclipse | teacher | learner | sound | accessible | need | tree | usage | audio | difference |

LSA

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | blind | child | blind\_learner | nonsighted | student | blind\_student | participant | story | sight\_programmer | tree |
| 1 | design | torino | audio\_programming\_language | auditory\_cue | accessible | barrier | source\_code | block | audiohighlight | structure |
| 2 | child | learning | skill | auditory | blockbased\_programming | computer\_science | story | sight\_programmer | blind\_programmer | solution |
| 3 | environment | design | novice\_blind\_learner | computer\_program | block | data | audio | blind | area | nonsighted |
| 4 | program | inclusive | solve\_problem | debugger | blockly | project | navigate | blockbased\_programming\_language | tree | audiohighlight |
| 5 | student | computational | program | program | user | tree | structure | blind\_programmer | reading | description\_language |
| 6 | using | story | interact | debug | library | nonsighted | block | output | barrier | aid |
| 7 | learning | teacher | motivating | comprehension | computing | information | output | tangible | user | graphic\_interface |
| 8 | accessible | learner | thinking | programmer | blockbased\_programming\_language | program | trial | game | structure | include |
| 9 | user | physical | help | cue | environment | interview | visual\_programming\_language | blind\_visually\_impair | complete\_task | development |

LDA

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | program | student | base | learner | blind | nonsighted | accessibility | child | participant | language |
| 1 | blind | blocksbased\_  programming | tree | accessible | barrier | language | computer\_science | design | blind\_developer | game |
| 2 | using | user | exploratory | research | sight\_programmer | child | introductory | learning | blind\_programmer | graphic\_interface |
| 3 | design | feedback | finding | visually\_impair | blind\_student | environment | course | torino | screen\_reader | description\_language |
| 4 | environment | block | developer | challenge | computer\_science | auditory\_cue | java | user | blind | sound |
| 5 | auditory\_cue | include | approach | environment | blind\_programmer | design | incorporate | ability | skimming | development |
| 6 | development | identify | challenge | process | project | computer\_program | blind | inclusive | problem | accessible |
| 7 | programmer | environment | findings | software\_  developer | create | memory | important | tree | navigate | result |
| 8 | research | textbased | impairment | learning | information | computer | software | vision | evaluation | program |
| 9 | result | design | visually\_impair | blind | comprehension | sufficient | simple | accessible | structure | blind |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Topics | LDA | LSA | NMF |
| 1 | Tool helps in reading, skimming and navigating source code/ code structure | 8 | 6/8/9 | 0 |
| 2 | Design of an accessible and inclusive, collaborative environment for children using blocks | 7 | 1 | 1 |
| 3 | Audio programming Language for the blind |  | 2 | 2 |
| 4 | Adding accessibility using Auditory cue to aid code comprehension, navigation and debugging in programs environments | 0/ 5 | 0/3 | 3 |
| 5 | Accessible Block-based programming Environment / | 1/ 7 | 4 | 4 |
| 6 | Challenges blind programmers face in IDES | 2/3/8 |  | 5 |
| 7 | Introducing CS to blind students and Barriers they face. | 4/ 6 | 5 | 6 |
| 8 | Description language to aid blind programmers in guI | 9 | 9 | 7 |
| 9 | Tangible block-based programming with audio and stories |  | 7/6 | 8 |
| 10 | Comparison of comprehension techniques of sighted vs blind developers | 4/Not sure | 8 | 9 |
| 11 | Creating an accessible environment for learning programming/CS | 0 | 0 |  |

Topic 6 of LDA Cannot be classified

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PAPER | LDA | LSA | NMF | PAPER | LDA | LSA | NMF |
|  | 1/ Y | 0 Partially Y but should be 4 | 4 Y |  | 0 Y | 6 partially yes due to code comprehension and navigation | 3 Y |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  | N 2 key words found des & sr | N | Y |

The distance between circles shows how different the topics are from each other. You can see that circle 2 and 3 are overlapping. This is because of the fact that topic 2 (Eiffel Tower) and topic 3 (Mona Lisa) have many words in common such as "French", "France", "Museum", "Paris", etc. [20]

If you hover over any word on the right, you will only see the circle for the topic that contains the word. For instance, if you hover over the word "climate", you will see that the topic 2 and 4 disappear since they don't contain the word climate. The size of topic 1 will increase since most of the occurrences of the word "climate" are within the first topic. A very small percentage is in topic 3, as shown in the following image: [20]

|  |  |
| --- | --- |
| **Topic** | **Codes** |
| 0 | Code / skimming / tool / structure / participant / tactile / blind\_programmer / navigate / programmer / eclipse. |
| 1 | Child / learning / torino / design / inclusive / computational / physical, ability / vision / teacher. |
| 2 | blind\_learner / programming / audio\_programming\_language / skill / novice\_blind\_learner / solve\_problem / interact / motivating / thinking learner. |
| 3 | auditory\_cue / nonsighted / program / auditory / computer\_program / programmer / research / debugger / debug / sound. |
| 4 | blockbased\_programming / computing / environment / user / blockly / design / impact / foster / outreach / accessible. |
| 5 | challenge / software\_developer / blind\_developer / developer / face / difficulty / blind / ides / interview / need. |
| 6 | blind\_student / barrier / computer\_science / project / interview / create / information / goal / dissertation / tree. |
| 7 | solution / development / aid / blind / description\_language / graphic\_interface / blind\_programmer / programming / regard / usage. |
| 8 | story / block / accessible / blockbased\_programming\_language / tangible / game / student / programming / output / audio |
| 9 | sight\_programmer / code / blind\_programmer / blind / area / reading / method / prioritize / comprehension difference. |