# MOOCs RECOMMENDER BASED ON LEARNING STYLES

Software Requirement Specification

Project ID: 18-036

Author: Porawagama A.S

IT 14142024

Bachelor of Science Special (Honors) in Information Technology

Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

Date of Submission: 2018-05-08

# MOOCs RECOMMENDER BASED ON LEARNING STYLES

Project ID: 18-036

Author: Porawagama A.S

IT 14142024

Supervisor: Mr. Nuwan Kodagoda

Date of Submission: 2018-05-08

## DECLARATION

I hereby declare that the project work entitled “MOOCs Recommender Based on Learning Styles” (Topic Modelling and Transcript Complexity components) submitted to the Sri Lanka Institute of Information Technology, is a record of original work done by our group under the guidance of Mr. Nuwan Kodagoda (Supervisor) and Ms. Kushnara Suriyawansa (Co- Supervisor), and this project work is submitted in the fulfillment for the award of the Bachelor of Science (Special Honors) in Information technology Specialization in Software Engineering. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma. The diagrams, research results and all other documented components were developed by myself and I have cited clearly any references I have made.

Name: Porawagama A.S Signature:

# TABLE OF CONTENTS

## LIST OF FIGURES



## LIST OF TABLES

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this document is to present a detailed description of requirements for “Topic Modelling and Transcript Complexity” Components of our research project, MOOCs Recommender Based on Learning Styles. The document will explain the purpose and features, the interfaces, what the system will do and the constraints under which it must operate. All parts are intended primarily for software engineers, building or maintaining the software.

## 1.2 Scope

This document is intended for the developers of the system. This document covers the requirements for the “Topic Modelling and Transcript Complexity” of the proposed “MOOCs Recommender Based on Learning Styles” application. The application will work as a platform to provide the most suitable, relevant and optimal learning resources to the learners based on their preferred style of learning. The “Topic Modelling” component deals with discovering the abstract “topics” that occur in a collection of courses and the “Transcript Complexity” component will try to identify the language complexity of the course. The main objective of Topic modeling is such that it can provide essential information when mapping MOOCs with the key-words search by the user. The objective of Transcript Complexity is to suggest the best option to the learners based on their language knowledge level. This document will focus on addressing the requirements to develop the component.

## 1.3 Definitions, Acronyms and Abbreviation

| | |
|---|---|
| MOOCs | Massive Open Online Courses |
| LDA | Latent Dirichlet Allocation |
| OS | Operating System |
| RAM | Random Access Memory |
| SRS | Software Requirement Specification |

## 1.4 Overview

The final product is targeted to be used by any learner who wants to learn using MOOCs. The learners can use the system to find the most appropriate and suitable MOOC courses based on their learning styles. Topic Modelling and Transcript Complexity are two sub-components of the overall system and its main objectives are to discover abstract topics of a course and calculate the complexity of the course.

This SRS document intends to cover all the functional and non-functional requirements of the Topic Modelling and Transcript Complexity Components. Each of them has been discussed clearly in detail. All are described under three chapters. The first chapter provides an overall description of the components. The purpose and scope of the SRS is also mentioned.

The second chapter will provide an in-depth overview of the functionalities as focusing on developers. The section will focus on comparing the software application to the existing systems. It then moves on to explain several interfaces, constraints and operation of users to provide the reader with a better perspective of understanding the product. The chapter then explains about the summary of major functions, characteristics of users, constraints of system, assumptions and dependencies and finally conclude with apportioning of requirements.

The third chapter includes the specific requirements of the system and will be written primarily for developers. This section will describe the technical aspects and in-depth detail of the functionalities mentioned in the previous chapter. Both sections of the document describe the same software product in its entirety but are intended for different audiences and thus use different languages. The document will conclude with providing any supporting information regarding the content of the document.

# 2 OVERALL DESCRIPTIONS

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body.
In MOOCs platform, transcripts are available for each video lecture. Specific topics can be extracting from the document which is useful in the scenario where the learner wants to search for MOOC with specific topics. Phrase mining technique will be used to get short texts from the massive texts. Then for topic modeling, Nouns only approach will be used since it is the most relevant for topic suggesting as it has given a better result for the approach.

Similarly, determining complexity of the language in transcript can be another important factor for filtering a MOOC by the learner’s language knowledge level. Various linguistic features will be used to measure the complexity where features can be: syntactic, semantic or discourse.

## 2.1 Product Perspective

Topic Modelling and Transcript Complexity are not directly used in systems. They are mostly sub-components of an overall system.

For Topic modeling, Latent Dirichlet allocation (LDA) is a technique which automatically discovers topics in documents. For example, given a set of documents, assume that there are some latent topics of documents that are not observed. Each document has a distribution over these topics. For instance, suppose the latent topics are 'loops', 'data types', 'functions', 'conditions'. Then a document may have the following distribution over the topics: 50% loops, 40% data types, 8% functions, 2% conditions. Another document might have a different distribution over the topics. Also, for each topic, you have a distribution over the words in the vocabulary. For example, for loops topic, the probability of word 'while loop' would be higher than that of 'string'. For data types, 'string' will have higher probability than 'while loop', and so on. Now, a document is assumed to be generated as follows: first, we select a distribution over the topics, say 50% loops, 40% data types, 8% functions, 2% conditions, as above. We draw a topic from this distribution, say that

comes to be data types. Then, we draw a word from the distribution over words corresponding to data types topic. This is the first word of the document. We repeat this process for all words.

We estimate the topic distributions and the distribution of words for each topic during training. Now given a new document, we can generate the most likely distribution over the topics that generated the document.

Linguistic complexity plays an important role in how people process information, when it comes to online courses it becomes the core foundation of how the learners are going to grab the knowledge from the course. The complexity can be calculated by using three main elements.

- Syntactic complexity - A few simple measures, such as the average word length and the average sentence length, are used as rudimentary metrics of syntactic complexity.
- Semantic complexity - Type/token ratio is the simplest way to determine the lexical characteristics of the text. The metric represents the ratio of the unique words in the text to all words (text length)
- Content complexity - High content complexity score indicates a uniform distribution of named entities throughout the text, implying a higher complexity of the content presented in the text.

### 2.1.1 System Interfaces

- Database connectivity interface
- Web Scraping interface

### 2.1.2 User Interfaces

Topic Modelling and Transcript Complexity components are backend processes. Hence, no interfaces to interact with the user.

### 2.1.3 Hardware Interfaces

No special hardware devices are needed other than the usual PC or Laptop.

### 2.1.4 Software Interfaces

- scikit-learn
- Jupyter Notebook
- Google Tensorflow
- Keras – Deep Learning Library

### 2.1.5 Communication Interfaces

The user's laptop/desktop should have data communication methods such as 3G (HSPDA) or 4G (LTE).

### 2.1.6 Memory Constraints

- RAM of 4 GB or higher

### 2.1.7 Operations

Functionalities of the Topic Modelling and Transcript Complexity Components are carried out in the backend and hence, no operations are required by the user.

### 2.1.8 Site Adaption Requirements

Since the user can be of any nationality the user interface must be created for English language.

## 2.2 Product Functions

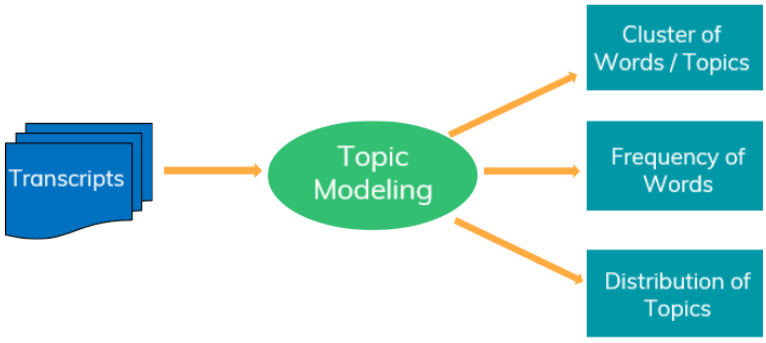


*Figure 1: Topic modeling component Architecture*

- ***Topic modelling***

The Latent Dirichlet allocation (LDA) is a technique that automatically discovers topics that these course transcripts contain. There are three main steps in the technique.

Step 1

Tell the algorithm how many topics there might be. We can either use an informed estimate (e.g. results from a previous analysis), or simply trial-and-error. In trying different estimates, we may pick the one that generates topics to our desired level of interpretability, or the one yielding the highest statistical certainty (i.e. log likelihood).

Step 2

The algorithm will assign every word to a temporary topic. Topic assignments are temporary as they will be updated in Step 3. Temporary topics are assigned to each word in a semi-random manner (according to a Dirichlet distribution, to be exact). This also means that if a word appears twice, each word may be assigned to different topics. Note that in analyzing actual documents, function words (e.g. “the”, “and”, “my”) are removed and not assigned to any topics.

Step 3 (iterative)

The algorithm will check and update topic assignments, looping through each word in every document. For each word, its topic assignment is updated based on two criteria:

How prevalent is that word across topics?
How prevalent are topics in the document?

The process of checking topic assignment is repeated for each word in every document, cycling through the entire collection of documents multiple times. This iterative updating is the key feature of LDA that generates a final solution with coherent topics.

- ***Linguistic complexity***

The Linguistic complexity can be measured using the three main variables. So, we can measure those three separately.

- Syntactic complexity

There are 14 measures of syntactic complexity.

1. Mean length of clause (MLC)
2. Mean length of sentence (MLS)
3. Mean length of T-unit (MLT)
4. Mean number of clauses per sentence (C/S)
5. Mean number of clauses per T-unit (C/T)
6. Mean number of complex T-units per T-unit (CT/T)
7. Mean number of dependent clauses per clause (DC/C)
8. Mean number of dependent clauses per T-unit (DC/T)
9. Mean number of coordinate phrases per clause (CP/C)
10. Mean number of coordinate phrases per T-unit (CP/T)
11. Mean number of T-units per sentence (T/S)
12. Mean number of complex nominals per clause (CN/C)
13. Mean number of complex nominals per T-unit (CN/T)
14. Mean number of verb phrases per T-unit (VP/T)

Following would be the procedure for calculating the syntactic complexity.

Input: plain English text

Step 1: Parsing using Stanford parser

Step 2: Retrieving & counting occurrences of

- Words, sentences, clauses, dependent clauses
- T-units, complex T-units
- Coordinate phrases, complex nominals, verb phrases

Step 3: Computing ratios for the 14 measures

Output: 14 syntactic complexity indices

- Semantic complexity

Semantic complexity correlates with the number of ways meaning can be derived and interpreted from an utterance. It also is associated with the types of syntactical structures necessary for it to be an intelligible utterance, and the number of different pathways meaning can be retrieved from (how easy or difficult it is to prompt for recall).

- Content complexity

The uncertainty here is associated with the information in the text. Named entity recognition algorithms can use to extract entities from a text (people, geographic locations, and organizations)

### 2.2.1 Use Case Diagram

Topic Modelling and Transcript Complexity components are backend processes. Hence, no interfaces to interact with the user.

### 2.2.2 Use Case Scenarios

| | |
|---|---|
| Use case Name | Identify abstract topics of course transcripts |
| Pre –Condition | Course transcripts are downloaded |
| Post-Condition | Collection of abstract topics are available |
| Actor | Backend System |
| Main Success Scenarios | 1. Select the transcript file from the system.<br>2. Run LDA algorithm and identify abstract topics.<br>3. The use case ends with successfully identifying abstract topics for the given document. |
| Extension | 1a. Invalid document selected.<br>1b. Select a valid source file and proceed. |

*Table 1: Use Case Scenario 1*

| | |
|---|---|
| Use case Name | Identifying linguistic complexity of transcript |
| Pre –Condition | Course transcripts are downloaded |
| Post-Condition | Percentage scores for the given transcript |
| Actor | Backend System |
| Main Success Scenarios | 1. Select the transcript file from the system.<br>2. Identify syntactics, semantics and content complexity.<br>3. The use case ends with successfully giving a score for each variable. |
| Extension | 1a. Invalid document selected.<br>1b. Select a valid source file and proceed. |

*Table 2: Use Case Scenario 2*

### 2.2.3 Activity Diagram

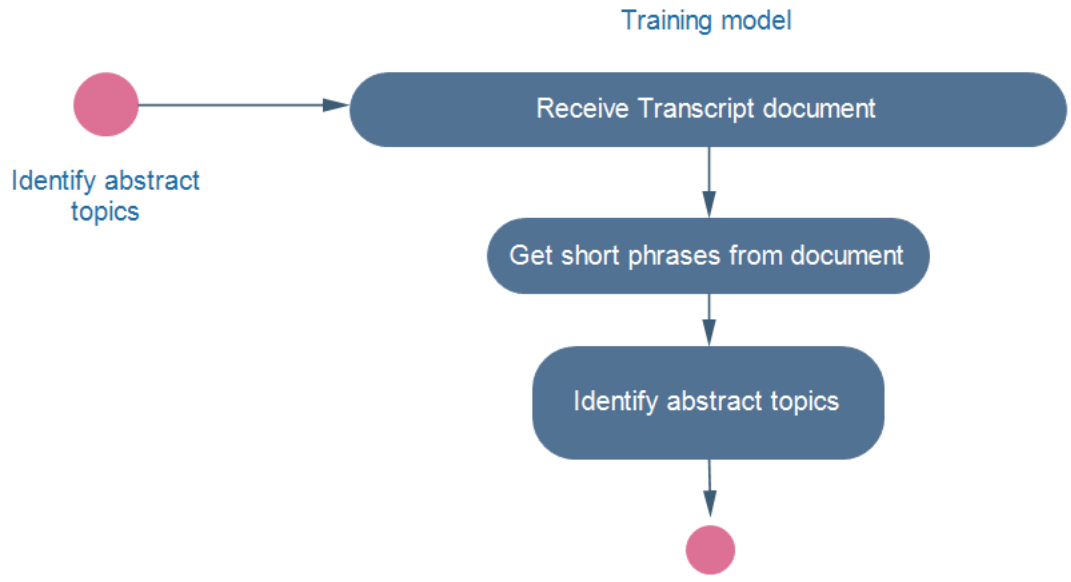


*Figure 2: Topic modeling activity diagram*

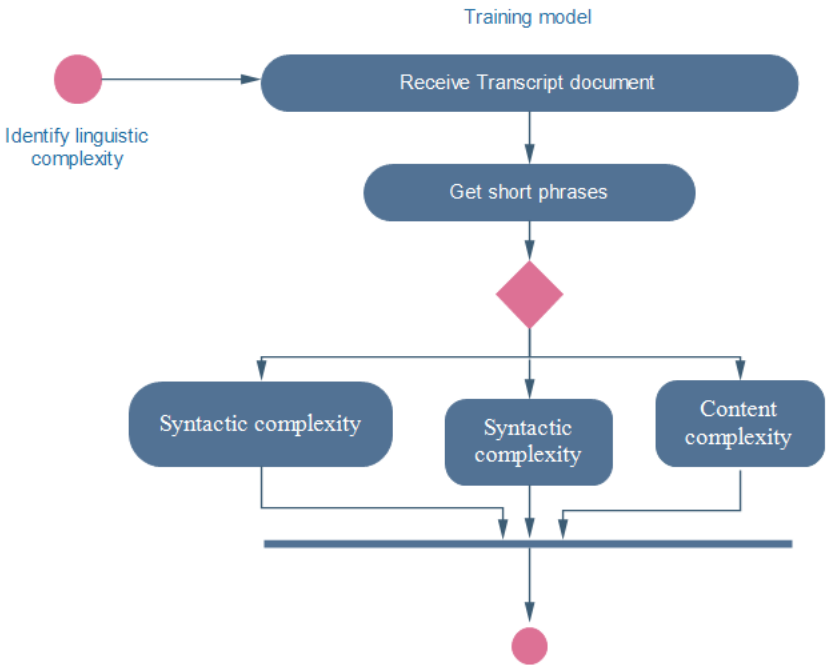


*Figure 3: Linguistic complexity activity diagram*

### 2.3 User Characteristics

Topic Modelling and Transcript Complexity components deals with the backend process and involves developing system to get abstract topic list and a score for the linguistic complexity of the transcript. It also consists of initial pre-processing steps and calculations at the later part, the user should be a software professional with appropriate knowledge of Data mining and other machine learning tools and technologies including basic mathematics.

### 2.4 Constraints

The backend process of Topic Modelling and Transcript Complexity involves some iterative computational tasks. Hence there are no heavy computational tasks, zero identified constrains.

### 2.5 Assumptions and Dependencies

- The operating system for the Topic Modelling and Transcript Complexity components are selected to be Windows.
- The internet connectivity can be established with ease on request.

### 2.6 Apportioning of Requirements

Essential Requirements:

1. Take a MOOC transcripts by course basis.
2. Perform topic modeling get a list of abstract topics.
3. Calculate the complexity of the transcripts.

Desirable Requirement:

1. Visualize the results graphically to generate analysis and reports.

# 3 SPECIFIC REQUIREMENTS

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

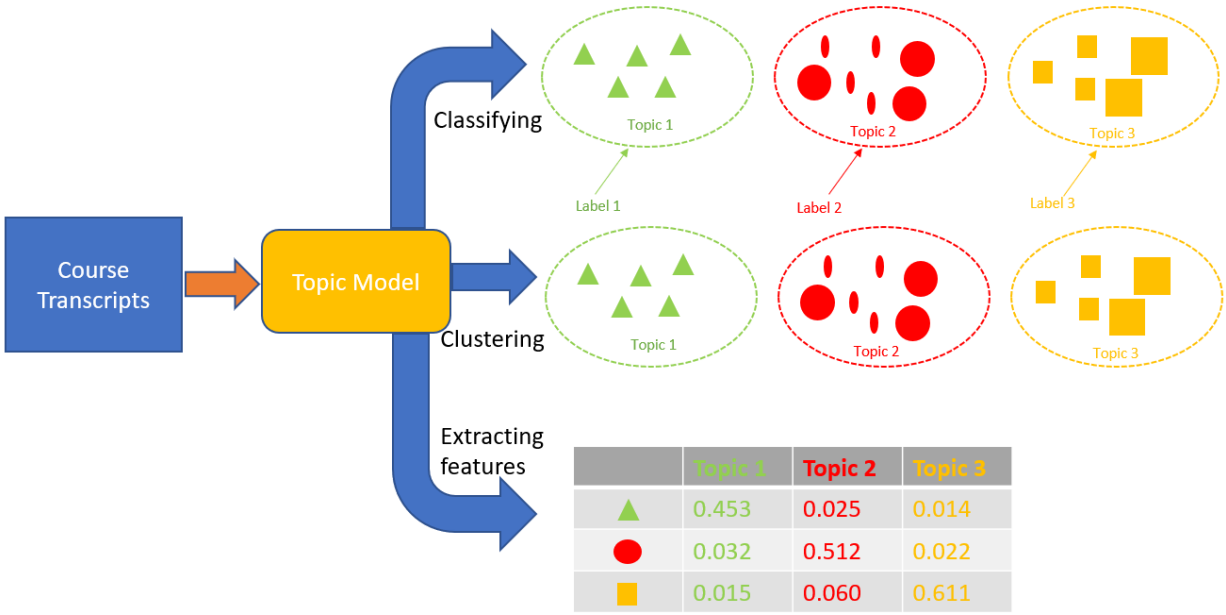


*Figure 4: Topic modelling to extract abstract topics*

The above figure gives a brief idea regarding the overall process of topic modelling. Given a course transcript, it first classifies it into given topics and label them, then it clusters the topics and give a score.

### 3.1.2 Hardware Interfaces

- Random Access Memory (RAM) – It can be used to speed up the computational tasks of training topic model and calculating the complexity.

### 3.1.3 Software Interfaces

- TensorFlow: It is a computational framework for building machine learning models. It provides wide variety of toolkits that allows the users to construct models at the preferred level of abstraction.
- Scikit-learn: It is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines.

### 3.1.4 Communication Interfaces

The backend service along with the database is hosted on the cloud. Hence, the communication between them requires internet connectivity. The connection can range from minimum of 3G to fast 4G (LTE).

## 3.2 Classes/Objects

Detailed class requirements can vary depending on the approach in implementation.

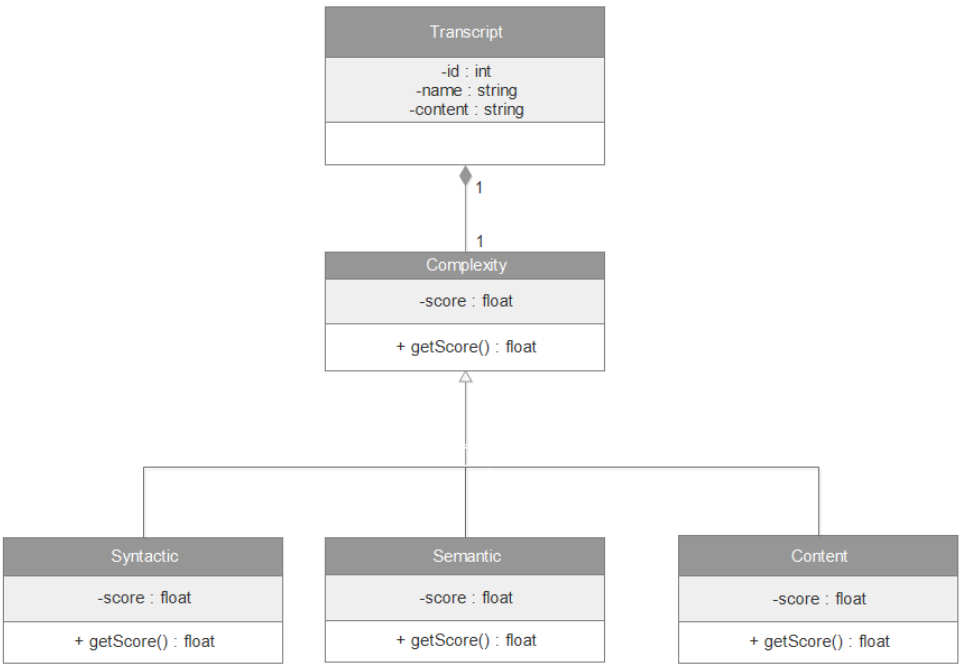


*Figure 5: Class diagram for linguistic complexity*

### 3.3 Performance Requirements

It is expected that the Topic Modelling and Transcript Complexity components will perform all the requirements stated under the product functions section. Some performance requirements identified are:

- Time taken to identify short phrases in the course transcript
- Time taken to iterate through all the sentences

### 3.4 Design Constraints

- MOOC course transcripts are only from three platforms: edX, Futurelearn and Coursera are considered.

### 3.5 Software System Attributes

#### 3.5.1 Reliability

This component should be able to perform and maintain its functions continuously without any hindrances. Given that the course transcripts are available, it should be reliable enough with the probability of around 99% to carry out its intended operations under any circumstances.

#### 3.5.2 Availability

As the backend process is hosted on the cloud server, the downtime may depend on the provider's services.

#### 3.5.3 Security

The topic model and complexity model are developed using standard coding standards such that it can be easily affected by outside attacks. Necessary steps are also taken into consideration for preventing the loss of statistical information generated during the process.

### 3.5.4 Maintainability

The topic modeling and complexity model are designed and developed such that it can be re-used with new transcript documents. It will help the model to become more robust to changes in the future.

## 3.6 Other Requirements

- Reusability: The specified component should be generic such that it should be suitable for use in other applications and scenarios as well.
- Interoperability: This component should be able to operate successfully by communicating with other components of the system such as: web scraping component. Similarly, it should be also efficient to exchange information with external components when needed.

# 4 REFERENCES


[1] K. Kotani and T. Yoshimi, “A Machine Learning Approach to Measurement of Text Readability for EFL Learners Using Various Linguistic Features,” US-China Educ. Rev. B, vol. 6, pp. 767–777, 2011..

[2] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, “Automated Phrase Mining from Massive Text Corpora,” IEEE Trans. Knowl. Data Eng., pp. 1–1, 2018.

[3] A. Sajid, S. Jan, and I. A. Shah, “Automatic Topic Modeling for Single Document Short Texts,” in 2017 International Conference on Frontiers of Information Technology (FIT), 2017, pp. 70–75.

[4] “Complete Guide to Topic Modeling.” [Online]. Available: https://nlpforhackers.io/topic-modeling/. [Accessed: 12-May-2018].

[5] “What is a good explanation of Latent Dirichlet Allocation?.” [Online]. Available: https://www.quora.com/What-is-a-good-explanation-of-Latent-Dirichlet-Allocation. [Accessed: 12-May-2018].

[6] “Analysis of Linguistic Complexity in Professional and Citizen Media” [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/1461670X.2017.1305285. [Accessed: 12-May-2018].

# 5 APPENDIX

```
from sklearn.decomposition import NMF, LatentDirichletAllocation, TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer

NUM_TOPICS = 10

vectorizer = CountVectorizer(min_df=5, max_df=0.9,
                        stop_words='english', lowercase=True,
                        token_pattern='[a-zA-Z\-][a-zA-Z\-]{2,}')
data_vectorized = vectorizer.fit_transform(data)

# Build a Latent Dirichlet Allocation Model
lda_model = LatentDirichletAllocation(n_topics=NUM_TOPICS, max_iter=10,
learning_method='online')
lda_Z = lda_model.fit_transform(data_vectorized)
print(lda_Z.shape)  # (NO_DOCUMENTS, NO_TOPICS)

# Let's see how the first document in the corpus looks like in different topic spaces
print(lda_Z[0])
```

```
def print_topics(model, vectorizer, top_n=10):
    for idx, topic in enumerate(model.components_):
        print("Topic %d:" % (idx))
        print([(vectorizer.get_feature_names()[i], topic[i])
                    for i in topic.argsort()[:-top_n - 1:-1]])

print("LDA Model:")
print_topics(lda_model, vectorizer)
print("=" * 20)
```