**Document analysis and Topic modeling on US security exchange commission company’s annual fillings.**

## 10-K Topic modeling project:

Finding out the most important keywords (topics in form 10-K).

Approaches used:

1. Latent Dirichlet allocation algorithm (using Gensim and sklearn approach).
2. Latent semantic allocation.
3. Visualization of key topics using pyldavis kit.

* Summarization: Generated the extractive summary of the document.
* Key phrase extraction using Rake
* Combining the results of key phrases and extractive summary. Drawn some interactive visualization using plotly. Visualization for list of companies discussing on key phrases and vice vesra.
* Built Ontology using protégé editor generated RDF’s and OWL2 (Web ontology language) for company’s security fillings. Used these ontology for general question answering problems.



**Requirement file**

        Please find the attached topic modelling folder of 10-K documents, the folder consists below information.

* **Input\_CIK** folder consist of CSV file that contain CIK no of the company.
* **Extract\_document\_CIK** is a python file that process all CIK No and generated output of all 10-K extracted URL of the last year (2017) store into Extracted\_Annual\_filing.csv file.
* **Extract.py** is a python file that process 10-K URL and generate a separate csv output file into Output folder, that file contain all the before cleaning sentences and after cleaning sentences.
* **LDA\_Sklearn\_RAKE\_Vizualization.ipynb**  Jupiter note book that  process file from Output folder and we are cleaning the sentence and generate the phrase with the help of (**RAKE**) module of the python and clean the corpus (stop words , lemma form, remove punctuation, emails..)and pass into  the LDA algorithm for topic modelling and visualize it , visualization results into Data visualization folder.
* Result of phrases and dominant topic of that phrases store into topics\_from\_phrases folder consists csv file that contain phrases and corresponding topic modelling keywords.



**Please see below List of python package required for run topic modelling: -**

|  |
| --- |
| **Spacy** |
| **rake\_nltk** |
| **pyLDAvis** |
| **glob** |
| **gensim** |
| **configparser** |
| **Beautiful Soup** |
| **requests** |
| **pandas** |
| **warning** |

# Topic Modeling on 10K Document

* How we are getting the best topics out of 10K documents?
* What is the approach we are following?
* Quantitative measure of topic model performance.

We are passing the extracted sentences to LDA model. This model performs below steps for generating the topics.

1. User will provide the CIK number csv file.
2. Based on this file, program will Search for all the 10K documents on the Securities and Exchange Commission (SEC) website for each CIK number.
3. Program will Extract the text from all the document based on start and end string provided and save it in another csv file.
4. Thereafter it will clean the data by removing the Email id's/stop words/spacing.
5. Performing the Lemmatization process for reducing the corpus size.
6. Creating the Dictionary and Corpus needed for Topic Modeling.
7. Applying **Latent Dirichlet Allocation (LDA)** algorithm: LDA represents documents as **mixtures of topics** that spit out words with certain probabilities. LDA is a bag-of-words model.
8. Diagnose the model performance with **perplexity and log-likelihood.**
9. Finding the matching phrases from the corpus based on the extracted topics by LDA. For this we are using the RAKE functionality (Rapid Automatic Keyword Extraction algorithm using NLTK).
10. As a result, we are getting the most important topics (list of keywords) and their associated phrases from the document.

**Latent Dirichlet Allocation (LDA)**

Suppose you have the following set of sentences:

* I like to eat broccoli and bananas.
* I ate a banana and spinach smoothie for breakfast.
* Chinchillas and kittens are cute.
* My sister adopted a kitten yesterday.
* Look at this cute hamster munching on a piece of broccoli.

What is latent Dirichlet allocation?  It’s a way of automatically discovering **topics** that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like

* **Sentences 1 and 2**: 100% Topic A
* **Sentences 3 and 4**: 100% Topic B
* **Sentence 5**: 60% Topic A, 40% Topic B
* **Topic A**: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, … (at which point, you could interpret topic A to be about food)
* **Topic B**: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, … (at which point, you could interpret topic B to be about cute animals)

The question, of course, is: how does LDA perform this discovery?

# **LDA Model**

In more detail, LDA represents documents as **mixtures of topics** that spit out words with certain probabilities. It assumes that documents are produced in the following fashion: when writing each document, you

* Decide on the number of words N the document will have (say, according to a Poisson distribution).
* Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics). For example, assuming that we have the two food and cute animal topics above, you might choose the document to consist of 1/3 food and 2/3 cute animals.
* Generate each word w\_i in the document by:
  + First picking a topic (according to the multinomial distribution that you sampled above; for example, you might pick the food topic with 1/3 probability and the cute animals topic with 2/3 probability).
  + Using the topic to generate the word itself (according to the topic’s multinomial distribution). For example, if we selected the food topic, we might generate the word “broccoli” with 30% probability, “bananas” with 15% probability, and so on.

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

**Log likelihood**

The most common way to evaluate a probabilistic model is to measure the log-likelihood of a held-out test set. This is usually done by splitting the dataset into two parts: one for training, the other for testing. For LDA, a test set is a collection of unseen documents wd, and the model is described by the topic matrix Φ and the hyperparameter α for topic-distribution of documents. The LDA parameters Θ is not taken into consideration as it represents the topic-distributions for the documents of the training set and can therefore be ignored to compute the likelihood of unseen documents. Therefore, we need to evaluate the log-likelihood

**L(w)=logp(w|Φ,α)=∑d logp(wd|Φ,α).**

of a set of unseen documents wd given the topics Φ and the hyperparameter α for topic-distribution θd of documents.

**Perplexity**

Likelihood of unseen documents can be used to compare models; higher likelihood implying a better model. The measure traditionally used for topic models is the \textit{**perplexity**} of held-out documents wd defined as

**Perplexity (test set, w) =exp{−L(w)/count of tokens}**

which is a decreasing function of the log-likelihood L(w) of the unseen documents wd; **the lower the perplexity, the better the model.**

However, the likelihood p(wd|Φ,α) of one document is intractable, which makes the evaluation of L(w), and therefore the perplexity, intractable as well.

**RAKE (Rapid Automatic Keyword Extraction algorithm)**

RAKE short for Rapid Automatic Keyword Extraction algorithm, is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text.

# Semantic Web Ontology

Ontologies have two major interests:

1. They can be used to perform logical inferences for deducing new facts, with a reasoner, and
2. They can link together various pieces of knowledge from different ontologies in the Semantic Web. The W3C (World Wide Web Consortium) proposed **OWL (Web Ontology Language)** for formalizing ontologies, and OWL ontologies are often represented as RDF graphs **(Resource Description Framework)**.

Many methods and tools have been proposed for the design, the edition, the maintenance, the alignment or the evaluation of ontologies, including the **Protégé** editor and the **HermiT** reasoner. Thus, it is a common need to interface ontologies with these programming languages.

Three strategies have been proposed for ontology programming interfaces:

1. Query languages such as SPARQL,
2. APIs (Application Programming Interfaces) such as OWLAPI, and
3. Ontology-oriented programming

**Ontology-oriented programming**:  It is based on object-oriented programming, a well-known and successful paradigm. It takes advantage of the similarities that exist between ontologies and object models, classes, properties and *individuals in an ontology correspond to classes*, attributes and instances in an object model. Ontology-oriented programming tries to connect, or even to unify, ontologies and the object model of a given programming language, e.g. a class in the ontology becomes a class in the programming language.

In practical terms, developing an ontology includes:

1. Defining classes in the ontology, arranging the classes in a taxonomic (subclass–superclass) hierarchy,
2. Defining slots and describing allowed values for these slots, filling in the values for slots for instances.
3. We can then create a knowledge base by defining individual instances of these classes filling in specific slot value information and additional slot restrictions.



# Software's for financial analysis which analyze SEC filings

Below are the tools available in the market working on Financial analysis, But none of the tools are providing the complete analysis on all financial documents like (10K, 10Q or 11K) which is freely available.

1. **Intrinio:** It provides the web API’s for data analysis on financial documents. This API provides the meta data for 10K like (filing\_date , accepted\_date , filing\_url etc..)

<https://www.api.intrinio.com/>

<https://youtu.be/rSWnldkvzH0>

1. **Rank and filed:** This is a web application which facilitates the list on financial documents based on Domain and companies. It will provide the complete list of financial documents submitted by any organization.

<http://rankandfiled.com/>

A fully-automated system that

1. received filings electronically,
2. facilitated the timely dissemination of filings to the public, and
3. 'tagged' data within each filing, allowing for the automatic screening and selection of filings for review (freeing both investors and Commission staff from tedious hunting and gathering).
4. **Filling Summary**: Web application that facilitates Summary of whole Form 10K document and gives the relevant companies.

<https://filingsummary.com/>

1. No specific tools for financial analysis by **Forbes.**
2. **Bloomberg** offers some services for Financial accounting. But are not freely available.

<https://www.bna.com/financial-accounting/>

1. **Faceset** provides multiple services for portfolio data management but **none of the services are free** they are charging some fees.

<https://www.factset.com/services/portfolio-data-management>

# Most relevant (Neighboring) sentences from phrases.

Here we are passing the phrases from the 10k document and fetching the most relevant or neighboring sentences from number of companies.

**Overview**

* Take the domain specific Phrase. Assumption is phrase length should be enough to explain the context. Choose minimum 5 phrases from each domain.
* Our model will find out the similarity between the phrases and sentences by providing the score.
  + For this we are using the Word2Vec model we are creating the corpus for training the word2vec model from 10k documents of 300 different companies. Bigger the corpus size more accuracy in the results.
  + Then calculate the vector of each phrase in the phrase.txt file and calculate the vector of each sentence from the passed 10k document.
* Consider more weightage for matching words from word embedding in phrase and sentence.
  + Next, we are finding the 50 most similar words of each word from phrase and comparing it with 50 most similar word from each sentence.
    - If the surrounded word from phrase would be same as the surrounded word from sentence, by taking the intersection. We are using this count of common surrounding words while calculating the cosine similarity score.
  + Finding the cosine similarity score between the phrases and the sentences.
    - Considering the surrounded word and multiply the cosine similarity score by the number of words matching (after taking the intersection of the phrase (surrounded words vs sentence surrounded words).
* Based on the similarity score we are choosing the top 5-10 sentences from each 10-K document.

Next, we are going to generate below report:

* From the generated report decide the threshold score depicting the matching sentence.
* If the score is above the threshold then we count that sentence considering it as the nearest neighbor of the phrase.
* Take the count of matching sentence for each company.
* **Report format**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Top 5 Matching Sentence | | | | | | |
|  | Company1 | | Company2 | | Company3 | |
| Phrases | Sentence | Score | Sentence | Score | Sentence | Score |
| p1 | s1 | score1 | s1 | score1 | s1 | score1 |
| p1 | s2 | score2 | s2 | score2 | s2 | score2 |
| p1 | s3 | score3 | s3 | score3 | s3 | score3 |
| p1 | s4 | score4 | s4 | score4 | s4 | score4 |
| p1 | s5 | score5 | s5 | score5 | s5 | score5 |
| p2 | s1 | score1 | s1 | score1 | s1 | score1 |
| p2 | s2 | score2 | s2 | score2 | s2 | score2 |
| p2 | s3 | score3 | s3 | score3 | s3 | score3 |
| p2 | s4 | score4 | s4 | score4 | s4 | score4 |
| p2 | s5 | score5 | s5 | score5 | s5 | score5 |

* **Final Report:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Companies discussion on phrases** | | | | |
| Company | Phrase1 | Phrase2 | Phrase3 | Phrase4 |
| Company1 | 2 | 3 | 0 | 1 |
| Company2 | 0 | 2 | 1 | 0 |
| Company3 | 2 | 1 | 0 | 1 |

1. To include the vector size as well as the similar word for score calculation.
2. Generate a matrix of Phrase vs sentence cosine score.
3. Apply PCA for matrix standardization. Using 2 components.
4. Find the row-wise maximum to get the final similarity score.