Non supervised text clustering with interpretability

Authors
Kevin Ferin
Yang Wang
Hicham Elouatiki
Guillaume Lehericy

July 24, 2020

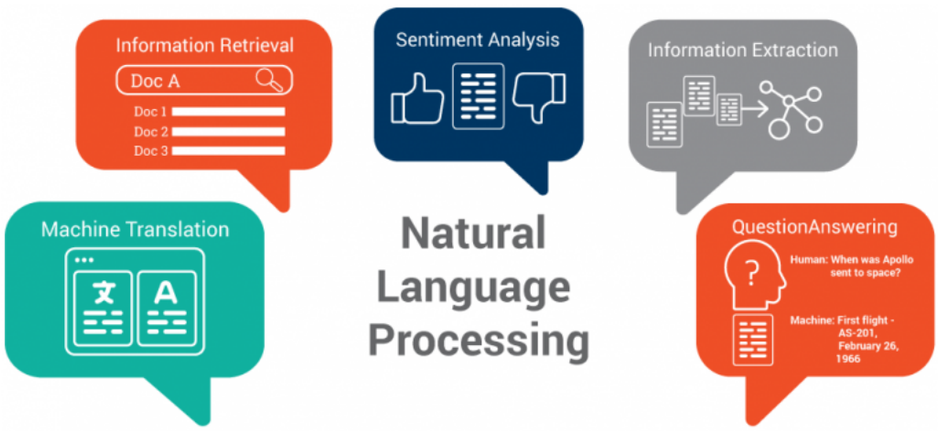
Contents

Abstract

This document summarizes the year long work on the main project that was completed during the Big Data master of Telecom Paris. The project is a cooperation between master's degree, researchers in the NLP and IBM.

1 Context

Natural Language processing and AI technology for businesses are not only increasingly popular but also inevitable topics for most companies. It has the power to automate support, enhance customer experiences, and analyze feedback. NLP describes the interaction between human language and computers. Human interaction is the driving force of most businesses. That means that there are countless opportunities for NLP to step in and improve how a company operates. This is especially true for large businesses that want to keep track of, facilitate, and analyze thousands of customer interactions in order to improve their products or services. It would be nearly impossible for employees to log and interpret all that data on their own, but technologies integrated with NLP can help to do it. It is in this optic that IBM tries to use NLP in his day to day process.



2 IBM and the NLP problematic

2.1 Who is IBM?

IBM (International Business Machines) ranks among the world's largest information technology companies, providing a wide spectrum of hardware, software and services offerings. IBM got its start in hardware and prospered in that business for decades, becoming the top supplier of mainframe computers. Over the years, the company shifted its focus from hardware to software and services. By the 2010s, IBM further modified its business mix to emphasize such fields as cloud-based services and cognitive computing. IBM Watson, a cognitive system, has become the company's high-visibility offering in the latter technology segment.

IBM was founded by Herman Hollerith in the late 1800s. Their first large contract was to provide tabulating equipment for the tabulation and analysis of the 1890 US census. The company grew quickly and, in the early 1920s the name was changed to IBM. IBM was the world leader in providing computer systems for both business and scientific applications. When, in 1964, they revolutionized the industry by bringing out the first comprehensive family of computers (the System/360) it caused many of their competitors to either merge or go bankrupt, leaving IBM in an even more dominant position. The advent of smaller computers, and IBM's failure to compete effectively in this field initially, caused some financial problems but IBM remains a major force in the industry. The services IBM offers are:

- Server hardware
- Storage
- Software
- Services
- Cloud
- Cognitive offerings

2.2 Why the NLP?

This technology represents the possibility of increasing the efficiency of some IBM's processes:

Volume : a large email/feedback client dataset daily generated.

Analysis : an analyst having the job knowledge has to extract information from the dataset (detect entities/sentiments/purposes/etc.).

Labeling : then documents are labeled by arrival order so we can dispatch them in the corresponding service.

This work represents an heavy charge for a company services. NLP should be able, theoretically, to help this process. A preprocessing of the data will reduce the time analysis spent manually by the analysts by offering them some annotations and key words propositions.

3 How to read this paper?

This paper is a collaboration between the 4 authors and represents the work done between beginning of November 2019 and the June 2020. The paper mostly addresses the research done by the team in order to map the unsupervised clustering field of NLP.

After some first steps to obtain an overview on the subject (all the sect 4. of the paper) by running some test and displaying some first results, we went into the depth of internet to identify more advanced papers/research/methods. The section 5. addresses this work by showing and explaining the relevant tools we will/may use in the following steps of the project. Since different tools are using -based on- similar methods, you will find some redundancy between distinct sub-sub-section. From section 6. you will find the final methods applied to the dataset to obtain the unsupervised clustering with the results.

4 Classic Clustering Methods

To have a first overview of the work load to realise to solve the problematic, we decided to apply different clustering analysis methods on some corpus dataset.

4.1 ENRON dataset

In order to fit the theoretical dataset format of IBM, we chose to work on the famous ENRON emails data-set. IBM wants to apply NLP on interactions between people. We judged that emails are closed to this format by providing both simple messages like a chat (internal communication between employees) and more complex one like full description of issues or projects. So why ENRON? First because it is one of the few collections of emails that are publicly available with a large quantity of data. Enron was the World's Leading Energy Company which went bankrupt in 2001. The dataset is from the different investigations that followed. The corpus contains 517,431 messages grouped by their 150 users. Each folders contain sub-folders which represent the schema of the user mail box. The email format is more problematic for direct analysis. It contains some headers (”Date”,”To”,”Cc”,etc.). The content is separated with the headers by a blank line, followed by the signature and quotation if they exist.

```
Message-ID: <12400760.1075857437225.JavaMail.evans@thyme>
Date: Wed, 30 Aug 2000 04:39:00 -0700 (PDT)
From: tysongs@aol.com
To: fsturm@enron.com
Subject: Gregg Tyson's Resume
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: TYSONGS@aol.com
X-To: fsturm@enron.com
X-cc:
X-bcc:
X-Folder: \Fletcher_Sturm_Dec2000\Notes Folders\Junk
X-Origin: Sturm-F
X-FileName: fsturm.nsf

It was good talking to you.  If you find any opportunity over there, please
let me know.

 - Gregg Tyson Resume.doc
```

4.2 Cleaning and PreProcessing

Organizational email dataset presents several problems. Some are less relevant in our analysis, like multiple identities for the same user (a user can be referred by several addresses, names, IDs, however the different formats are referring to the same person in the system). Since the purpose of the project is to cluster the text, the identity of sender/receiver does not impact us (we may have to come back on it if we decide later to include identities into the clustering method). The second issue is more important: duplicate emails. Duplicate emails can increase the bias of the method (will be explained later in the technical part). To remove duplicate emails we used the MD5 digest of the bodies. Doing so allow us to remove a large amount of the data (first tries show a final dataset of 250,484 emails). However we are losing some information about possible receivers. Since the first steps do not included an analysis about users, we can afford to lose these information. Another issue is generic emails created automatically like meeting, recall, etc. For the first clustering test, we did not apply further cleaning than removing duplicates and small body emails (less than X words). These consideration will be applied later if needed. Next, simple text transformations are applied to obtain the relevant text for the corresponding corpus. We use packages like nltk for stopwords, WordNetLemmatizer, string for punctuation and some others to remove names from text.

4.3 Vectorize

Now, all documents are represent by a word vector. Each vector's size is different since the raw texts coming from the emails have also different size. In order to process with the analysis, we will modify these vectors to obtain fixed size ones. Two methods will be used:

4.3.1 Count Vectorizer

Our model, as the most applied ones to text dataset, is using the words occurrence account in a document. A convenient way to represent a document is to use a vector called Bag-of-Words (BoW) containing the frequency of each words (unrelated to their order) in the document.

If we consider all the words appearing in our M training document that we name V (Vocabulary), we can create **an index**, which is a bijection associating each words w to an whole number. The index represents its position in V

For a single document extracted from a set of documents containing $N = |V|$ different words, a BoW representation will be a vector with a size of N, whose value at its index position w will be its frequency inside the document. All documents will be translated to a single vector with an unique size N. These vector's size are variant since each document contains only a few set of the words existing in the full dataset.

4.3.2 Tf-idf Vectorizer

Another method which can be applied is the Term Frequency - Inverse document frequency one (TF-IDF). This method allows to highlight the word's w significance inside the document d relatively to the all corpus. To explain in detail the formula applied to the frequency computation, it is important to precise that we got M training documents. The full dataset size is noted D and $T \in \mathcal{R}^{NxM}$ is corresponding to the BoW output.

First, we compute the frequency of each words w inside the document $d \in D$:

$$TF(T, w, d) = \frac{T_{w,d}}{\sum_{w'=1}^{|V|} T_{w',d}}$$

Then, the inverse frequency of w to see how representative the word is compared to its occurrence in the corpus:

$$IDF(T, w) = \log\left(\frac{M}{|\{d : T_{w,d} > 0\}|}\right)$$

Finally, to compute the result, we multiply these two previous values and we obtain the importance of w in the current document relatively to the corpus:

$$TF - IDF(T, w, d) = TF(X, w, d) \cdot IDF(T, w)$$

Now that we got a logic representation of each document by a fixed size vector, we can try to dispatch these documents in different groups called cluster according to their meanings.

4.4 Clustering algorithm - Latent Dirichlet Allocation (LDA)

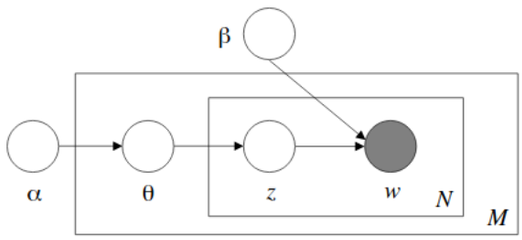
The Latent Dirichlet Allocation (LDA) algorithm was first published in 2003 and the authors are Blei, Ng, and Jordan, 2003 . It matters to notice that this algorithm is not restricted to clustering but can also be used in text classification. The ability to cluster our documents in different group is what import here. Moreover, we are able to extract from it the relevant words which help us to interpret the results in a user point of view. This clustering interpretation is an obligation in the method applied since we need to advise the user about the text label (IBM requirement). Using the same notation:

- We have a corpus D with M documents composed by w words. The dataset generate by all the words is a vocabulary list V.We note $N = |V|$ the number of unique words.
- Each document $d \in D$ is a words combination $(w_1, ..., w_k)$

The input is the matrix word-per-document $T \in \mathcal{R}^{|V|xN}$ that we generate with Tf-idf. One of the hypothesis is that some of the documents dealing with same topics are using a similar set of words. The other hypothesis is that the data is generated in a specific way. Here the rules on how words are chosen:

1. Choose $N \sim Poisson(\xi)$
2. Choose $\theta \sim Dir(\alpha)$
3. For each of the N words w_n :
   (a) Choose a topic $z_n \sim Multinomial(\theta)$
   (b) Choose a word w_n from $P(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

In few lines, the documents are represented by a topic distribution probabilities, and topics by a words distribution probabilities. In order to identify more clearly the link between parameters we can visualize it:



This representation displays the 3 different layers and the parameters. α and β are the corpus parameters, θ parameter belongs to a specific document, therefore it is unique to each of them. At last *zandw* are words parameters. The same way, z is unique to each words.

What matters is to obtain the variables probabilities according to a document. It will allow us to determine the topics distribution z through the corpus documents (θ parameter) and the key words they are composed with (*parameter*):

$$p(\theta, z|d, \alpha, \beta) = \frac{p(\theta, z, d|\alpha, \beta)}{p(d|\alpha, \beta)} \tag{1}$$

We can now write the joint probability of a topic distribution θ. A topic and its words list d are representing a type of document according to their parameters α and β, as shown below thanks to the words/data generation hypothesis:

$$p(\theta, z, d|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{2}$$

Then we sum all z and we apply the integral on alphas to get:

$$p(d|\alpha, \beta) = \int p(\theta|\alpha) \Big(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \Big) d\theta \tag{3}$$

Without explaining the mathematics details, we can notice that these probabilities cannot be computed so we have to approximate them. Many methods exist to solve this issue, and as much papers have been written about the subject. The most commons are:

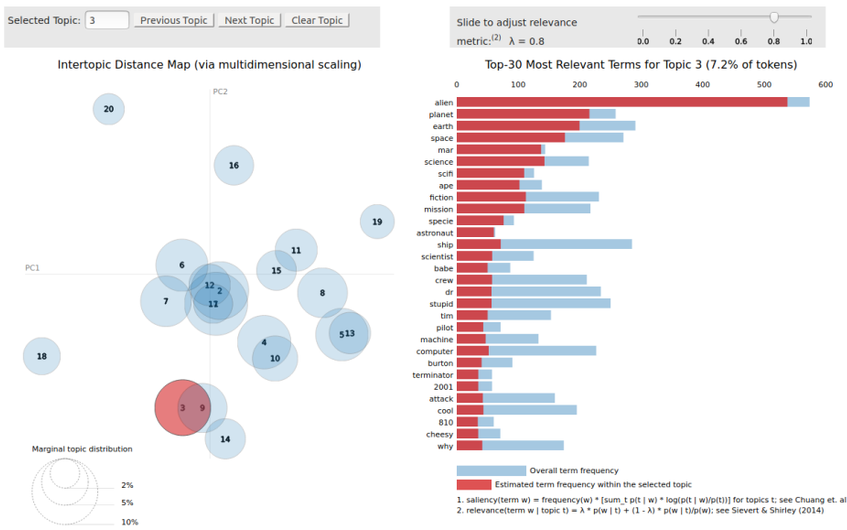
- **Markov chain Monte Carlo (MCMC)** is a method quite easy to understand and unbiased. It intends to find the posterior probability distribution by randomly drawing values from a complex distribution of interest. MCMC are named that way, because the previous sampled values (previous states) affect the generation of the next random sample value (in other words, the transition probabilities between sample values is a function of the most recent sample value.)
- **Variational inference** can be seen as deterministic alternative to MCMC and is as accurate but faster than the previous method. Variational Inference replaces MCMC's random, somewhat independent sampling, with optimization. It seeks to optimize a simplied parametric distribution to be close in Kullback-Leibler divergence to the posterior. The goal is to find a point in the subfamily distribution that is the closest to $P(z|w)$. Similarity is measured using Kullback–Leibler divergence, which is a non-symmetric measure of the difference between two probability distributions P and Q.

In practice the MCMC method can be summarized in few steps :

1. Randomly assign each word in each document to one of the K topics
2. For each document d
   (a) Assume that all topics assignments except for the current one are correct
   (b) Calculate two proportions
      i. Proportions of words in document d that are currently assigned to topic z : $p(z|d) \approx p(z|\theta) * p(\theta|\alpha)$
      ii. Proportions of assignments to topic z over all documents that come from this word w : $p(w|z) \approx p(w|z, \beta)$
   (c) Multiply those two proportions and assign w a new topic based on that probability $p(z|d) \cdot p(w|z)$
3. After a number of iterations we will reach a steady state where assignments make sense

4.5 Visualisation

One of the main issue when dealing with topic modelling is visualizing the topics and the distributions of the documents in the space. A tool especially designed to visualize the results of an LDA model does exist. LDAvis: A method for visualizing and interpreting topics by C.Sievert and K.E. Shirley. This tool's purpose is to answer few questions about our topic model: (1) What is the meaning of each topic?, (2) How prevalent is each topic?, and (3) How do the topics relate to each other ?



This graphical representation is divided in 2 parts :

- The left panel presents a global view of the topic model and answers questions (2) and (3). The projections used is a PCA with a Jensen-Shannon
- The right panel is a horizontal barchart whose bars represent the individual terms that are the most useful for interpreting the selected topic on the left, and allows users to answer question 1, "What is the meaning of each topic?". A pair of overlaid bars represent both the corpus-wide frequency of a given term as well as the topic-specific frequency of the term.

It uses two measures to select which word appears in the right part :

- This first one is saliency firstly introduced by Jason Chuang, Christopher D. Manning and JeffreyHeer. 2012b. in the paper "Termite: Visualization Techniques for Assessing Textual Topic Models."
- The second one, introduced by the authors of this paper LDAvis, is the relevance. This metric is used to rank terms within topics and its formula is as follow : Given a weight parameter $0 \leq \lambda \leq 1$ and p_w is the marginal probability of w in the corpus,

$$r(w, z|\lambda) = \lambda log(p(w|z)) + (1 - \lambda) log(\frac{p(w|z)}{p_w})$$

  There is a slide bar to select different values of λ and pick which representation suits best our needs.

5 State of the Art

5.1 Representation of texts

5.1.1 Document as Bag-of-Words

The "Bag-of-Words" Assumption rather than "Part-of-Speech" Clustering textual data groups similar documents and reveals hidden connections. As textual data is more complex than numeric data, it requires to be treated differently. There are few assumptions for processing textual data, but different assumptions lead to different approaches. The "bag-of-words" assumption is one of the most popular ones. It considers a piece of text (or a document) as a set of words. In this assumption, the words' order is ignored, only their existence matters. There are other assumptions that believe the words' order conveys necessary information, which is taken as feature in the corresponding models. Depending on the assumption choice, clustering methods on textual data vary. In general, these methods can be categorized into distance-based algorithms, phrase-based algorithms, probabilistic generative models, textual streams methods and graph approaches.

5.1.2 Word embedding

Word Embedding is a type of word representation that allows words with similar meaning to be understood by machine learning algorithms. Technically speaking, it is a mapping of words into vectors of real numbers using the neural network, probabilistic model, or dimension reduction on word co-occurrence matrix. It is language modeling and feature learning technique. Word embedding is a way to perform mapping using a neural network. There are various word embedding models available such as word2vec (Google), Glove (Stanford) and fastest (Facebook). With word embedding, a word is represented as a dense vector (which is really different from the sparse representation of BoW). Specifically, it is a vector space model (VSM) which represents words in a continuous vector space such that words that share common contexts and semantics are located in close proximity to one another in the space.

There is two ways of doing so :

- **Count-based methods** : Count-based methods compute the statistics of how often some word co-occurs with its neighbor words, and then map these count-statistics down to a small, dense vector for each word. (ex: Latent Semantic analysis)
- **Predictive methods** : Predictive models directly try to predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model). Traditionally, predictive models are trained using the maximum likelihood principle to maximize the probability of the next words given the previous words in terms of a softmax function over all the vocabulary word (ex : word2vec)

5.1.3 Word2Vec

Word2Vec idea came into this world in 2013 as a two-layer neural networks which is trained to reconstruct linguistic contexts of words. It takes as its input a large corpus of words and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec is a particularly computationally-efficient predictive model for learning word embed dings from raw text.

Word2Vec is able to capture multiple different degrees of similarity between words, such that semantic and syntactic patterns can be reproduced using vector arithmetic. Patterns such as "Man is to Woman as Brother is to Sister" can be generated through algebraic operations on the vector representations of these words such that the vector representation of "Brother" - "Man" + "Woman" produces a result which is closest to the vector representation of "Sister" in the model.

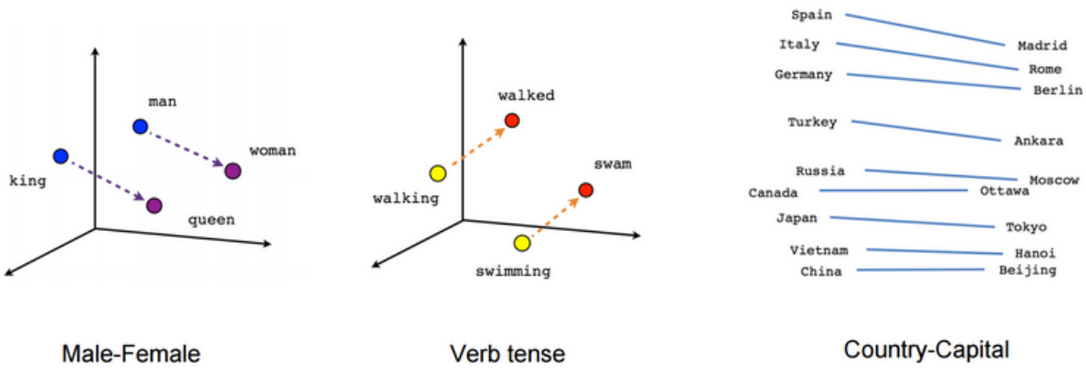


Figure 1: Example of semantic relationship between words in vector space

Now that we have seen what such word representation can give, let's dive into how this algorithm works. Actually there are two different models that you can see in Figure 2.

- **Continuous Bag-of-Words (CBOW)** : CBOW predicts target words (e.g. 'mat') from the surrounding context words ('the cat sits on the'). Statistically, it has the effect that CBOW smooths over a lot of the distributional information (by treating an entire context as one observation). For the most part, this turns out to be a useful thing for smaller data sets.
- **Skip-Gram** : Skip-gram predicts surrounding context words from the target words (inverse of CBOW). Statistically, skip-gram treats each context-target pair as a new observation, and this tends to do better when we have larger data sets.

The specific thing about this neural nets is that they are not used the way it is designed to. Word2Vec uses a trick, it is a simple neural network with a single hidden layer, and like all neural

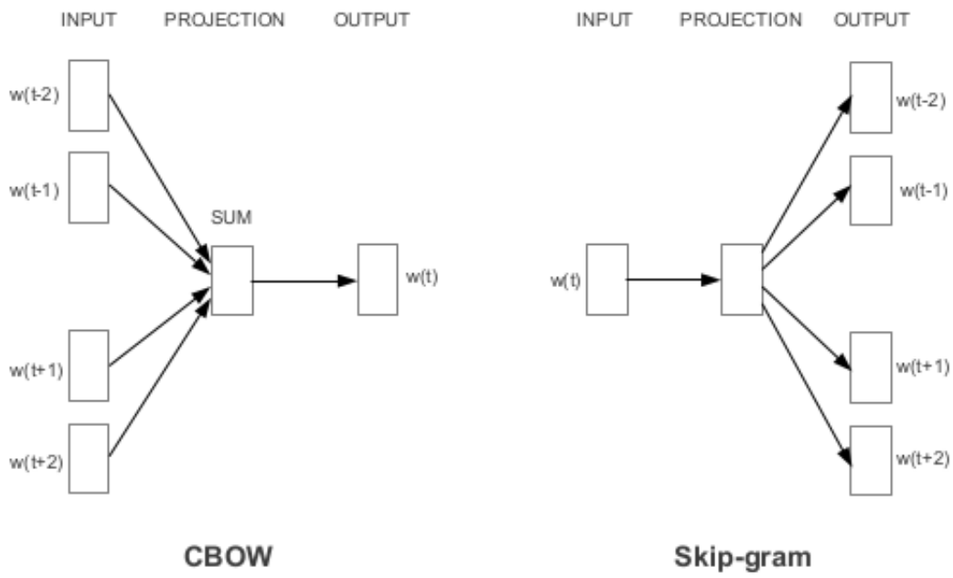


Figure 2: Visual representation of CBOW and skip-gram model for word2vec

networks, it has weights, and during training, its goal is to adjust those weights to reduce a loss function. However, Word2Vec is not going to be used for the task it was trained on, instead, we will just take its hidden weights, use them as our word embedding, and toss the rest of the model. The input layer takes one-hot vectors of each words and try to predict. So as the corpus grows, the vocabulary will also grow. Then there will be a lot of weight in the neural net and so training will be longer (exponential). A common technique is **Negative Sampling**. Training a neural network means taking a training example and adjusting all of the neuron weights slightly so that it predicts the training sample more accurately. In other words, each training sample will tweak all of the weights in the neural network. Our skip-gram neural network has a tremendous number of weights, all of which would be updated slightly by every one of our billions training samples. Negative sampling addresses this by having each training sample only modify a small percentage of the weights, rather than all of them. (reminder : the output is a one hot vector so the size of the vocab, the goal is to predict 1 on the good word and 0 on other. Later we refer as negative the words we need to assign 0).In a word, negative sampling updates not every weights at each iteration but a few of them selected randomly according to their probability of appearance. The higher the probability, the bigger chance to be selected as negative sample (ie the weights referring to its neuron will be updated). The probability of selecting a negative sample is computed according to the formula :

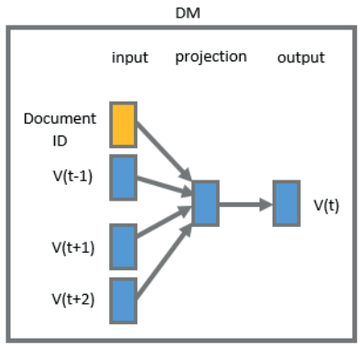
$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} f(w_j)^{\frac{3}{4}}}$$

5.1.4 Paragraph Vector With A Distributed Bag Of Words (PVDBOW)

.

They are variations of the **CBOW** Models and **Skip Gram** Models, which use to train Word2Vec, it as extending the idea of context to paragraphs by adding a label or a document-ID.

> **The Distributed-Memory Model** closely resembles the CBOW model of Word2vec. This model tries to predict a target word given its surrounding context words with the addition of a paragraph ID

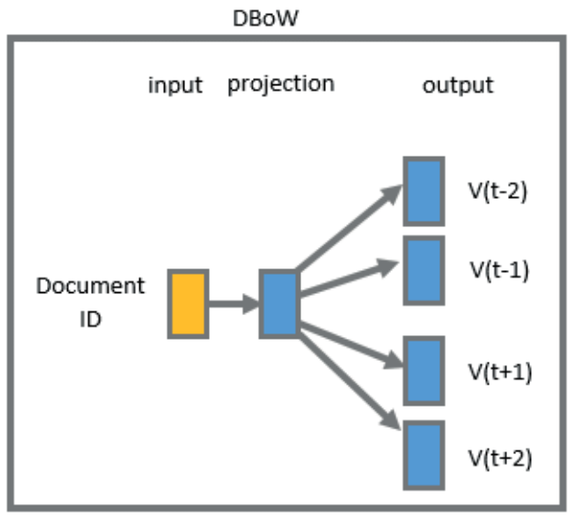


Distributed-Memory Model

Every paragraph (Document ID) is mapped to a unique vector, represented by a column in a matrix (denoted by D), as is each word in the vocabulary. The contexts are fixed-length and sampled from a sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph but not across paragraphs.
As in word2vec, vectors must be summarized in some way into a single vector; but unlike word2vec, the authors use concatenation in their experiments. Notice that this preserves order information. Similar to word2vec, a simple softmax classifier (in this case, actually hierarchical softmax) is used over this summarized vector representation to predict the task output. Training is done the standard way, using stochastic gradient descent and obtaining the gradient via back-propagation. Notice that only the paragraphs in the training corpus have a column vector from D associated with them. At prediction time, one needs to perform an inference step to compute the paragraph vector for a new paragraph: The document vector is initialized randomly. Then, repeatedly, a random word is selected from the new document, and gradient descent is used to adjust input-to-hidden-layer weights such that softmax probability is maximized for the selected word, while hidden-to-softmax-output weights are fixed. This results in a representation of the new document as a mixture of training corpus document vectors (i.e. columns of D), naturally residing in the document embedding space.

5.1.5 Paragraph vectors (Doc2Vec)

Sometimes referred to as Doc2Vec, this method, presented by Le Mikolov, 2014 is perhaps the first attempt to generalize Word2Vec to work with word sequences. In Doc2Vec, the name of the document, like file name or file ID will be the input, and the sliding window of the words from the document is the output.
Similar to Word2vec, the authors introduce two variants of the paragraph vectors model: **Distributed Memory Model Of Paragraph Vectors (PV-DM)** and **Distributed Bag of Words (PV-DBOW)** The second variant of paragraph vectors, despite its name, is perhaps the parallel of word2vec's skip-gram architecture; the classification task is to predict a single context word using only the paragraph vector. At each iteration of stochastic gradient descent, a text window is sampled, then a single random word is sampled from that window, forming the below classification task.



Distributed Bag-Of-Words Model

Training is otherwise similar, except for the fact that word vectors are not jointly learned along with paragraph vectors. This makes both memory and runtime performance of the PV-DBOW variant much better.

implementations and enhancements

Le and Mikolov, 2014 demonstrated the use of paragraph vectors on several text classification and sentiment analysis tasks, while Dai et al, 2015 examined it in the context of document similarity tasks and https://arxiv.org/pdf/1607.05368.pdfLau and Baldwin, 2016 benchmarked it against a forum question duplication task and the Semantic Textual Similarity (STS) SemEval shared task. Both later papers present an extended evaluation of the method (the former focusing on the PV-DBOW variant), comparing it to several other methods, and also giving practical advice (the later including code).

The method has a Python implementation, as part of the gensim package, and a PyTorch implementation Again,Lau Baldwin, 2016 also supplied the code used for their examination. Finally, various enhancements to the method have been proposed. For example, Li et al, 2016 extend the method to also incorporate n-gram features, while Thongtan Phienthrakul, 2019 suggest using cosine similarity instead of dot product when computing the embedding projection (also providing a Java implementation).

5.2 Mixing LDA and word embeddings - lda2vec

After learning about word embedding and its ability to catch the semantic and syntactic patterns in word and texts, we wondered if we could use this representation and mix it to our previous LDA model. Then we found out that Christopher Moody already had that idea in his paper about lda2vec in 2016.

Inspired by Latent Dirichlet Allocation (LDA), the word2vec model is expanded to simultaneously learn word, document and topic vectors. Lda2vec is obtained by modifying the skip-gram word2vec variant. In the original skip-gram method, the model is trained to predict context words based on a pivot word. In lda2vec, the pivot word vector and a document vector are added to obtain a context vector. This context vector is then used to predict context words. This idea of context vector is not new. Paragraph vectors, for example, also explored this idea in order to learn fixed-length representations of variable-length text fragments. In their work, for each text fragment (size of a paragraph) a dense vector representation is learned, similar to the learned word vectors.

The lda2vec model goes one step beyond the paragraph vector approach by working with document-sized text fragments and decomposing the document vectors into two different components. In the same spirit as the LDA model, a document vector is decomposed into a document weight vector and a topic matrix. The document weight vector represents the percentage of the different topics, whereas the topic matrix consists of the different topic vectors. A context vector is thus constructed by combining the different topic vectors that occur in a document. Consider the following example: in the original word2vec model, if the pivot word is 'French', then possible context words might be 'German', 'Dutch', 'English'. Without any global (document-related) information, these would be the most plausible guesses. By providing an additional context vector in the lda2vec model, it is possible to make better guesses of context words. If the document vector is a combination of the 'food' and 'drinks' topics, then 'baguette', 'cheese' and 'wine' might be more suitable. If the document vector is similar to the 'city' and 'geography' topics, then 'Paris', 'Lyon' and 'Grenoble' might be more suitable.

Note that these topic vectors are learned in word space, which allows for easy interpretation: you simply look at the word vectors that are closest to the topic vectors. In addition, constraints are put on the document weight vectors, to obtain a sparse vector (similar to LDA), instead of a dense vector. This enables easy interpretation of the topic content of different documents.

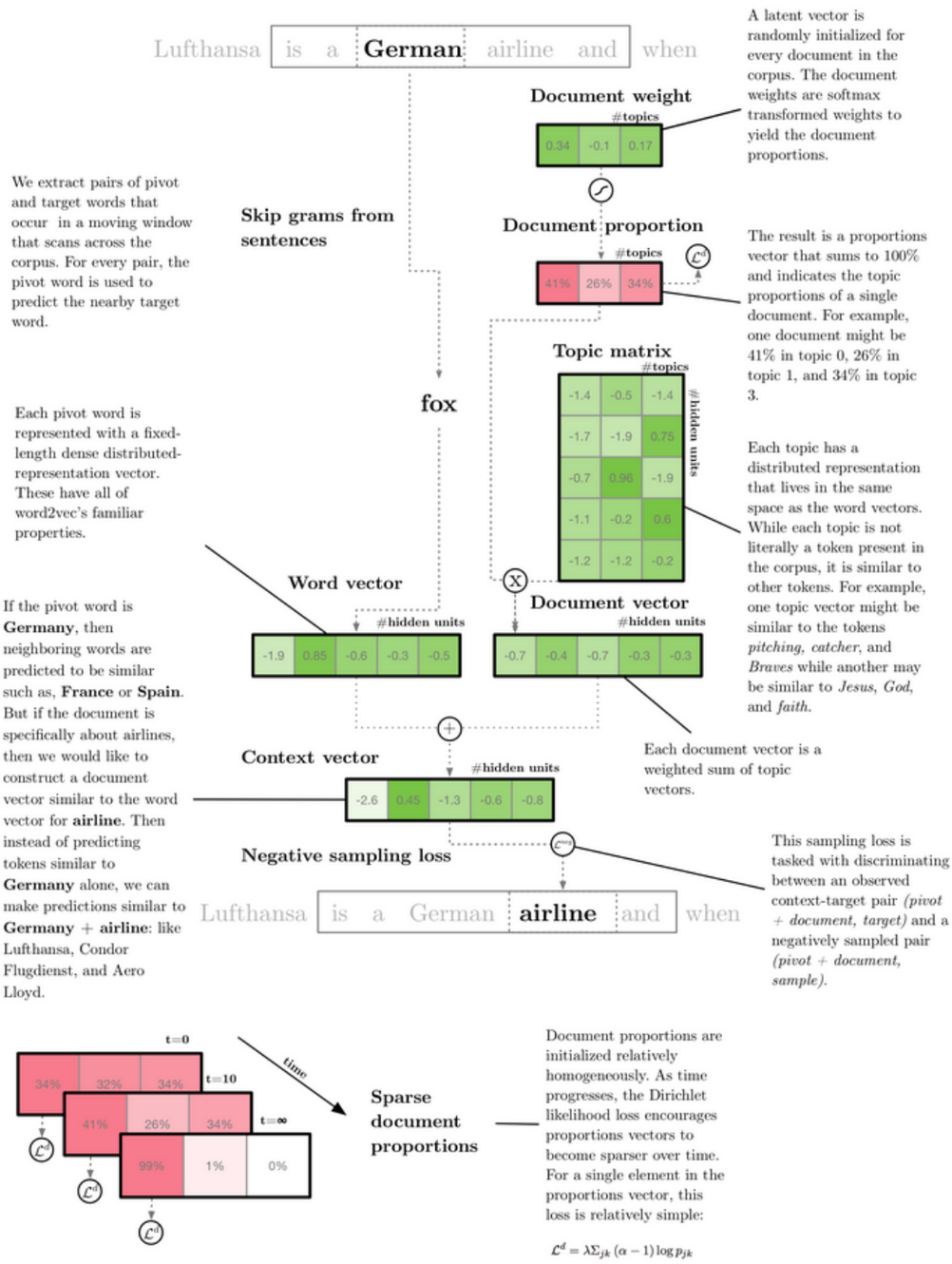


Figure 3: Visual representation of lda2vec

This algorithm thus try to minimize the sum of the loss function of the skip-gram's negative sampling $\mathcal{L}_{ij}^{neg}$ and a Dirichlet-likelihood termover document weights $\mathcal{L}^d$. The loss is computed using a context vector c_j, a pivot word w_j, a target word w_i, negatively-sampled word vector w_l, topics vectors t_i, document vectors d_j :

$$c_j = w_j + d_j$$

$$d_j = p_{j0} * t_0 + p_{j1} * t_1 + ... + p_{jn} * t_n$$

$$\mathcal{L} = \mathcal{L}^d + \sum_{i,j} \mathcal{L}_{ij}^{neg}$$

$$\mathcal{L}_{ij}^{neg} = log\sigma(c_j * w_i) + \sum_{l=0}^{n} log\sigma(-c_j * w_l)$$

$$\mathcal{L}^d = \lambda \sum_{j,k} (\alpha - 1) log(p_{jk})$$

That last equation is what gives us a sparse document vector d_j to keep the interpretability of LDA in a way that a document can't be about a lot of topics, usually a document is about between 1 and 3 topics.

5.3 Non-probabilistic models and probabilistic models

5.3.1 Non-probabilistic models

We tested in Jupyter notebook some non-probabilistic clustering models like K-means, hierarchical model, NMF. Spectral is also promising, mainly due to its capacity to reduce the dimension of document representation vectors.

5.3.2 New Probabilistic models

Probabilistic models are more adapted to our project, thus we are studying some specific topic models such as Biterm Topic Model, Multi-Grain Topic Model, Topic Modeling with Minimal Domain Knowledge, Author-Topic Model, Dynamic Topic Models, Embedded Topic Model.

A biterm topic model for short texts

Uncovering the topics within short texts, such as tweets and instant messages, has become an important task for many content analysis applications. However, directly applying conventional topic models (e.g. LDA and PLSA) on such short texts may not work well. The fundamental reason lies in that conventional topic models implicitly capture the document-level word co-occurrence patterns to reveal topics, and thus suffer from the severe data sparsity in short documents. In this paper, we propose a novel way for modeling topics in short texts, referred as biterm topic model (BTM). Specifically, in BTM we learn the topics by directly modeling the generation of word co-occurrence patterns (i.e. biterms) in the whole corpus. The major advantages of BTM are that 1) BTM explicitly models the word co-occurrence patterns to enhance the topic learning; and 2) BTM uses the aggregated patterns in the whole corpus for learning topics to solve the problem of sparse word co-occurrence patterns at document-level. We carry out extensive experiments on real-world short text collections. The results demonstrate that our approach can discover more prominent and coherent topics, and significantly outperform baseline methods on several evaluation metrics. Furthermore, we find that BTM can outperform LDA even on normal texts, showing the potential generality and wider usage of the new topic model.

Anchored Correlation Explanation: Topic Modeling with Minimal Domain Knowledge

"While generative models such as Latent Dirichlet Allocation (LDA) have proven fruitful in topic modeling, they often require detailed assumptions and careful specification of hyperparameters. Such model complexity issues only compound when trying to generalize generative models to incorporate human input. We introduce Correlation Explanation (CorEx), an alternative approach to topic modeling that does not assume an underlying generative model, and instead learns maximally informative topics through an information-theoretic framework. This framework naturally generalizes to hierarchical and semi-supervised extensions with no additional modeling assumptions. In particular, word-level domain knowledge can be flexibly incorporated within CorEx through anchor words, allowing topic separability and representation to be promoted with minimal human intervention. Across a variety of datasets, metrics, and experiments, we demonstrate that CorEx produces topics that are comparable in quality to those produced by unsupervised and semi-supervised variants of LDA." - https://arxiv.org/abs/1611.10277

5.4 Evaluation of clustering and Topic Model

How to assess the quality of a clustering? These questions are important for our project for two reasons. First, in the ideal world, the texts clustering results should be the same as a human classification, however, the human's criteria can be very subjective. A bank may want to classify emails by different business lines (mortgage, trading...) rather than different sectors(Europe,Asia...) or departments(marketing,compliance...). Secondly, an optimal clustering algorithm does not exist. In other words, different algorithms—or even different configurations of the same algorithm—produce different partitions and none of them have proved to be the best in all situations.

Thus, for a unsupervised performance evaluation we can compute different partitions and select the one that best fits the data.

The process of estimating how well a partition fits the structure underlying the data is known as cluster validation. Cluster validation is a difficult task and lacks the theoretical background other areas, such as supervised learning, have.

5.4.1 General measurement of unsupervised classification

Perplexity

The perplexity of a discrete probability distribution p is defined as

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

where H(p) is the entropy (in bits) of the distribution and x ranges over events. (The base need not be 2: The perplexity is independent of the base, provided that the entropy and the exponentiation use the same base.) This measure is also known in some domains as the (order-1 true) diversity.

However, recent studies have shown that predictive likelihood (or equivalently, perplexity) and human judgment are often not correlated, and even sometimes slightly anti-correlated. Optimizing for perplexity may not yield human interpretable topics.

5.4.2 Intrinsic Evaluation Metrics

Performance metrics of clustering:

- Silhouette index (Silm): This index is a normalized summation-type index. The cohesion is measured based on the distance between all the points in the same cluster and the separation is based on the nearest neighbour distance.
- Davies-Bouldin Index
- Calinski–Harabasz (CH): It is a ratio-type index where the cohesion is estimated based on the distances from the points in a cluster to its centroid.The separation is based on the distance from the centroids to the global centroid.

5.4.3 Evaluation of Topic Model - Topic Coherence

Probabilistic topic models, such as LDA, are popular tools for text analysis, providing both a predictive and latent topic representation of the corpus. However, there is a longstanding assumption that the latent space discovered by these models is generally meaningful and useful, and that evaluating such assumptions is challenging due to its unsupervised training process. Besides, there is a no-gold standard list of topics to compare against every corpus.

Nevertheless, it is equally important to identify if a trained model is objectively good or bad, as it is to be able to compare the different models/methods. To do so, one would require an objective measure for the quality. Traditionally, and still for many practical applications, to evaluate if "the correct thing" has been learned about the corpus, implicit knowledge and "eyeballing" approaches are used. Ideally, we'd like to capture this information in a single metric that can be maximized, and compared.

The special case of measuring coherence of topics has been recently studied to remedy the problem that topic models give no guaranty on the interpretability of their output. Several benchmark datasets were produced that record human judgements of the interpretability of topics. M. Röder, A. Both conduct a systematic search of the space of coherence measures using all publicly available topic relevance data for the evaluation(Exploring the Space of Topic Coherence Measures). Their results show that new combinations of components outperform existing measures with respect to correlation to human ratings.

Framework of Coherence Measures

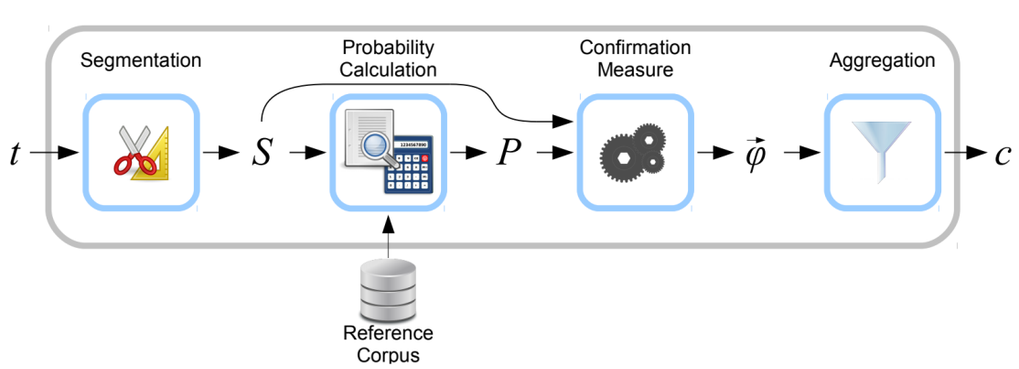


Figure 4: Overview over the unifying coherence framework—its four parts and their intermediate results

- **Segmentation of word subsets**

  Coherence of a word set measures the degree that a subset is supported by another subset. The result of the segmentation of a given word set W is a set of pairs of subsets of W. The

definition of a subset pair consists of two parts that are differently used by the following confirmation measures. The first part of a pair is the subset for which the support by the second part of the pair is determined. Most proposed coherence measures for topic evaluation compare pairs of single words, e.g., the UCI coherence. Every single word is paired with every other single word. Other subsets segmentation methods can be one-pre(a word only to the preceding and words), one-suc(a word only to the succeeding words), one-all(every single word to all other words of the word set),one-any(every single word to any other words of the word set). Röder define another segmentation as one-set(every single word to all other words of the word set), the difference with one-all is one-set compare a word with all the words in the word set.

- **Probability Estimation** The method of probability estimation defines the way how the probabilities are derived from the underlying data source. Boolean document (Pbd) estimates the probability of a single word as the number of documents in which the word occurs divided by the total number of documents. In the same way, the joint probability of two words is estimated by the number of documents containing both words divided by the total number of documents. This estimation method is called boolean as the number of occurrences of words in a single document as well as distances between the occurrences are not considered. UMass coherence is based on an equivalent kind of estimation. There are variations, namely the boolean paragraph (Pbp) and boolean sentence (Pbs). These estimation methods are similar to boolean document except instead of documents paragraphs or sentences are used respectively.

  Boolean sliding window (Psw) determines word counts using a sliding window. The window moves over the documents one word token per step. Each step defines a new virtual document by copying the window content. Boolean document is applied to these virtual documents to compute word probabilities. Boolean sliding window can capture some information about proximity between word tokens.

- **Confirmation Measure** A confirmation measure takes a single pair Si = (W' , W) of words or word subsets as well as the corresponding probabilities to compute how strong the conditioning word set W supports W0 . This could be done either directly or indirectly. The confirmation measures md, mr and ml are called difference-, ratio- and likelihood-measure. There, log-likelihood (mll) and log-ratio measure (mlr) are also defined— the last is the PMI, the central element of the UCI coherence. Normalized log-ratio measure (mnlr) is the NPMI. The log-conditional-probability measure (mlc) is equivalent to the calculation used by UMass coherence.

- **Indirect confirmation measures** Instead of directly computing the confirmation of Si = (W0 , W), indirect computation of confirmation assumes that given some word of W, direct confirmations of words in W0 are close to direct confirmations of words in W with respect to this given word. Thus, indirect confirmation computes similarity of words in W0 and W with respect to direct confirmations to all words. This is an important idea, since words who semantically support each other may seldom mentioned together in documents in the reference corpus, like two competing brands of garments, but their confirmations to other words like “fashion” or “clothes” do strongly correlate.

- **Aggregation** Finally, all confirmations of all subset pairs Si are aggregated to a single coherence score. Arithmetic mean (σa) are the most commonly used.

Comparison of different measures

The best performing coherence measure is C_v which combines the indirect cosine measure with the NPMI and the boolean sliding window.

According to Roder's study, when the coherence measures use probabilities derived from the Wikipedia instead of the corpus used for topic learning, there was a better correlations for the scores of all datasets.

In terms of runtimes, the most important component is the probability estimation. The fastest is the boolean document and boolean paragraph. The window based estimation methods has the highest runtimes, this is caused by the need of window sliding throughout the documents. Thus, the size of the window does not have much influence on the runtimes. Another important component is the segmentation. While the segmentation of a specific topic is very fast, it controls the number of confirmation values that have to be calculated.

5.5 Overview of Adaptive Resonance Theory (ART)

(ART) (Grossberg, 1976a,b, 1980, 2013)is a biologically-plausible theory of how a brain learns to consciously attend, learn and recognize patterns in a constantly changing environment. The theory states that resonance regulates learning in neural networks with feedback (recurrence). Thus, it is more than a neural network architecture, or even a family of architectures. However, it has inspired many neural network architectures that have very attractive properties for applications in science and engineering, such as being fast and stable incremental learners with relatively small memory

requirements and straight-forward algorithms (Wunsch II, 2009). In this context, fast learning refers to the ability of the neurons' weight vectors to converge to their asymptotic values directly with each input sample presentation. These, and other properties, make ART networks attractive to many researchers and practitioners, as they have been used successfully in a variety of science and engineering applications. ART addresses the problem of stability vs. plasticity. Plasticity refers the ability of a learning algorithm to adapt and learn new patterns. In many such learning systems plasticity can lead to instability, a situation in which learning new knowledge leads to the loss or corruption of previously-learned knowledge, also known as catastrophic forgetting. Stability, on the other hand, is defined by the condition that no prototype vector can take on a previous value after it has changed, and that an infinite presentation of inputs results in forming a finite number of clusters (Xu Wunsch II, 2009).
ART addresses this stability-plasticity dilemma by introducing the ability to learn arbitrary input patterns in a fast and stable self-organizing fashion without suffering from catastrophic forgetting. shows the classical ART clustering algorithms:

ART models

ART algorithms	Description
ART1	Can stably learn to categorize binary inputs.
ART2	Can learn to categorize analog patterns presented in an arbitrary order.
ART2A	This algorithm is the fast version of ART2.
ART3	Extends ART by incorporating 'chemical transmitters' to control the search process in a hierarchical ART structure.
ARTMAP	Can rapidly self-organize stable categorical mappings between M-dimensional input vectors and N-dimensional output vectors.
Fuzzy ART	Incorporating computations from fuzzy set theory into ART1.
Fuzzy ARTMAP	Can rapidly learn stable categorical mappings between analog input and output vectors.

5.5.1 Fuzzy ART

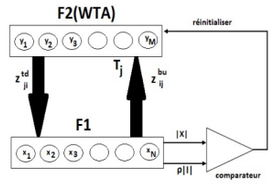
The fuzzy ART neural network was first introduced by Carpenter et al. in 1991 The fuzzy ART is an unsupervised learning algorithm, which is capable of learning in both off-line and on-line training modes. It is the most recent adaptive resonance framework, which provides a unified architecture for both binary and continuous value inputs.

The generalization of learning both analog and binary input patterns is achieved by replacing the appearance of the logical AND intersection operator $(\cap) in\ ART1\ by the\ MIN\ operator (\wedge)\ of\ the\ fuzzy\ set\ theory$(Pacellaetal., 2004).
$the fuzzy ART involves three main differences in comparison with ART1$:

There is a single weight vector connection.

Non-binary inputs can be processed.

In addition to vigilance threshold $(\rho), choice parameter(\alpha) and learning rate(\beta) should be determined.$



Topological structure of the fuzzy ART

We propose a new fuzzy clustering approach based on ART adaptive neural networks, Networks ART are competitive neural networks, made up of a set of neurons and which are the basis of model unsupervised learning.

They are able, from an autonomous structure, to develop stable clusters from arbitrary features.

Indeed ART only learns in its resonant state in which a prototype vector corresponding fairly closely to the current input vector is considered. In addition, ART has a self-checking structure allowing autonomous learning.

The approach we propose is a three steps:

- **Preprocessing phase** : During this data preprocessing phase, we cleans them so that they no longer have to deal with information irrelevant and redundant present or noisy and unreliable data. This phase includes tokenization, normalization and rooting.
- **Vector creation phase** : This phase makes it possible to form the vectors corresponding to each of the previously preprocessing phase documents.
- **Clustering Phase** :This phase takes as input the vectors resulting from the previous phase in order to proceed with clustering.In this phase the FuzzyART algorithm was used, which is an adaptive neural network that implements fuzzy logic in ART pattern recognition, thus improving generalization.

An optional feature of FuzzyART is complement coding, a means of incorporating the absence of features into model classifications, which goes a long way towards preventing the inefficient and unnecessary spread of categories. The similarity measures applied are based on L1. FuzzyART is known to be very sensitive to noise. During this phase FuzzyART is used for learning. In a second step a classifier Doc2Vec for the test.

6 BERT

6.1 A bit of context

BERT is a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks. It stands for Bidirectional Encoder Representations for Transformers. It has been pre-trained on Wikipedia and BooksCorpus and requires task-specific fine-tuning. We will dig deeper in how BERT actually works, what is new in this technology and why we decided to use it.

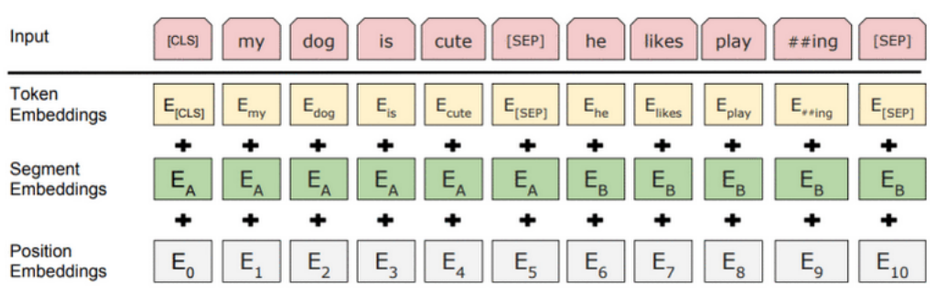
So to begin with we will try to understand what is the technology behind BERT. One of the biggest challenges in NLP is the lack of enough training data. Overall there is enormous amount of text data available, but if we want to create task-specific datasets, we need to split that pile into the very many diverse fields. And when we do this, we end up with only a few thousand or a few hundred thousand human-labeled training examples. Unfortunately, in order to perform well, deep learning based NLP models require much larger amounts of data — they see major improvements when trained on millions, or billions, of annotated training examples. To help bridge this gap in data, researchers have developed various techniques for training general purpose language representation models using the enormous piles of unannotated text on the web (this is known as pre-training). These general purpose pre-trained models can then be fine-tuned on smaller task-specific datasets, etc, when working with problems like question answering and sentiment analysis. This approach results in great accuracy improvements compared to training on the smaller task-specific datasets from scratch. BERT is a recent addition to these techniques for NLP pre-training; it caused a stir in the deep learning community because it presented state-of-the-art results in a wide variety of NLP tasks, like question answering.

The best part about BERT is that it can be download and used for free — we can either use the BERT models to extract high quality language features from our text data, or we can fine-tune these models on a specific task, like sentiment analysis and question answering, with our own data to produce state-of-the-art predictions.

6.2 How does Bert works ?

BERT relies on a Transformer (the attention mechanism that learns contextual relationships between words in a text). A basic Transformer consists of an encoder to read the text input and a decoder to produce a prediction for the task. Since BERT's goal is to generate a language representation model, it only needs the encoder part. The input to the encoder for BERT is a sequence of tokens, which are first converted into vectors and then processed in the neural network. But before processing can start, BERT needs the input to be massaged and decorated with some extra metadata:

- **Token embeddings:** A [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
- **Segment embeddings:** A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish between sentences.
- **Positional embeddings:** A positional embedding is added to each token to indicate its position in the sentence.



The input representation for BERT: The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Essentially, the Transformer stacks a layer that maps sequences to sequences, so the output is also a sequence of vectors with a 1:1 correspondence between input and output tokens at the

same index. And as we learnt earlier, BERT does not try to predict the next word in the sentence. Training makes use of the following two strategies:

- **Masked Language Model :** Bert works as a Masked Language Model but first let's dive into what is a Language Model. A Language Model is what an object which, given a sequence of words is able to predict the word that has the highest probability to appear right after that sequence. A Masked Language Model works the same but instead of predicting the next word, a number of words are masked and the LM predicts those words given previous AND next words into account. The idea here is "simple": Randomly mask out 15% of the words in the input — replacing them with a [MASK] token — run the entire sequence through the BERT attention based encoder and then predict only the masked words, based on the context provided by the other non-masked words in the sequence. However, there is a problem with this naive masking approach — the model only tries to predict when the [MASK] token is present in the input, while we want the model to try to predict the correct tokens regardless of what token is present in the input. To deal with this issue, out of the 15% of the tokens selected for masking: 80% of the tokens are actually replaced with the token [MASK], 10% of the time tokens are replaced with a random token and 10% of the time tokens are left unchanged.

  While training the BERT loss function considers only the prediction of the masked tokens and ignores the prediction of the non-masked ones. This results in a model that converges much more slowly than left-to-right or right-to-left models.

- **Next sentence prediction :** In order to understand relationship between two sentences, BERT training process also uses next sentence prediction. A pre-trained model with this kind of understanding is relevant for tasks like question answering. During training the model gets as input pairs of sentences and it learns to predict if the second sentence is the next sentence in the original text as well. As we have seen earlier, BERT separates sentences with a special [SEP] token. During training the model is fed with two input sentences at a time such that 50% of the time the second sentence comes after the first one, 50% of the time it is a a random sentence from the full corpus. BERT is then required to predict whether the second sentence is random or not, with the assumption that the random sentence will be disconnected from the first sentence. To predict if the second sentence is connected to the first one or not, basically the complete input sequence goes through the Transformer based model, the output of the [CLS] token is transformed into a 21 shaped vector using a simple classification layer, and the IsNext-Label is assigned using softmax.

The model is trained with both Masked LM and Next Sentence Prediction together. This is to minimize the combined loss function of the two strategies.

6.3 FlauBERT

6.3.1 From English to French

The first part of our project was done in english due to the facility to find NLP documentation. However IBM problematic is to find a model we could apply to a french corpus.To do so we stopped using the ENRON database and went to the CSL-french database. The Cross-Lingual Sentiment (CLS) dataset comprises about 800.000 Amazon product reviews in the four languages English, German, French, and Japanese. We decided just to take the test set (2000 reviews about books, 2000 about DVD and 2000 about music)to avoid too long computational time.

6.3.2 What is FlauBERT ?

A lot of the new ideas in Machine Learning and in other specific domain is done in English and BERT is exactly the same. As we saw earlier, BERT is learning a language model so it is specific a language which is generaly English. This paper, FlauBERT: Unsupervised Language Model Pre-training for French, introduced a way to pre-trained a BERT architecture on a French corpus so we could achieve the same very good results on French data.

6.3.3 Hugging Face architecture

As we saw previously, several pre-trained implementation of BERT are available on internet, we decided, in collaboration with IBM and out academic tutor we decided to use the Hugging Face pre-trained model of transformer BERT. But what is hugging face ?

Hugging Face is an NLP-focused startup with a large open-source community, in particular around the Transformers library. Transformers is a python-based library that exposes an API to use many well-known transformer architectures, such as BERT, RoBERTa, GPT-2 or DistilBERT, that obtain state-of-the-art results on a variety of NLP tasks like text classification, information extraction, question answering, and text generation.

Hugging face Transformers (formerly known as pytorch-transformers and pytorch-pretrained-bert) provides state-of-the-art general-purpose architectures (BERT, GPT-2, RoBERTa, XLM,

DistilBert, XLNet, CTRL...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

6.4 Architecture

Hugging face implemented several architectures of the FlauBERT paper. We decided to let the choice of the architecture to the final user. Here is a brief explanation of those :

- **FlauBERT-small :** 6-layer, 512-hidden, 8-heads, 54M parameters
- **FlauBERT-base :** 12-layer, 768-hidden, 12-heads, 137M parameters
- **FlauBERT-large :** 24-layer, 1024-hidden, 16-heads, 373M parameters

7 RAKE words extraction

RAKE also known as Rapid Automatic Keyword Extraction is a keyword extraction algorithm that is extremely efficient which operates on individual documents to enable an application to the dynamic collection, it can also be applied on the new domains very easily and also very effective in handling multiple types of documents, especially the type of text which follows specific grammar conventions.

In our model RAKE allow us to adapt easily the keywords extractions to any input structure. It is also necessary since we need to interpret the clusters created from a user point of view. The clusters had to be named by the user which means that he has to be able to get the meaning behind the corpus inside them.

The input parameters for the RAKE Algorithm comprise a list of stop words also a set of phrase delimiters and word delimiters. It uses stop words and phrase delimiters to partition the document into candidate keywords, these candidate keywords are mainly the words that help a developer in extracting the exact keyword necessary to get information from the document.

8 Clustering coherence

Recent studies have shown that predictive likelihood (or equivalently, perplexity) and human judgment are often not correlated, and even sometimes slightly anti-correlated.This limitation of perplexity measure served as a motivation for more work trying to model the human judgment, and thus Topic Coherence.

Topic coherence is a score measuring the degree of semantic similarity between high scoring words in the topic. This score help to distinguish between topics that are semantically interpretable and the ones created from only a mathematical point of view. To summarize, a high coherence inside a cluster means that the cluster is composed from texts sharing a same context (sport, finance, plane, etc...). The coherence scores is based on several mathematical/statistical principles:

- **one-set segmentation of the top words :** compares words to the total word set W using words context vectors.
- **a boolean sliding window :** determines word counts using a sliding window. The window moves over the documents one word token per step. Set to 1 if word appears, else 0.
- **an indirect confirmation measure :** compute the relation between a word and an other one. Indirect means that the formula takes into account the relation between two words according to their relation with all other words.
- **an aggregation :** vector aggregated to a single coherence score

The coherence model is explained here. In our tests it allows us to compare the difference between the models in an objective way. Coherence is a score between 0 and 1, it is considered good if above 0.5 but below 0.9. A too high coherence means that the cluster is unlikely due to the variety of words composing a text.

9 Library Implementation - SCBert

9.1 Why a python library ?

Our projects consisted in providing a tool that could be used by IBM when they are labelling their clients's dataset. In order to be able to test our tool on datasets close to the real application we asked IBM to have an access to at least one data with labels but due to confidentiality with their clients they could not provide us. Moreover the tool should be used easily without install problems. Lastly the users may not always know very well python so the tool must be well documented.

Thus to answer all those conditions we decided to develop a python library that we called **SCBert**. This python library is published on the Python Package Index (PyPi) so it is available to anyone with an internet connection through a simple :

pip install SCBert

The source code of our library is also available on Github here. All the functions are documented, which means the inputs are detailed regarding their type and their purposes. What the functions do is also explained when hovering the function in a python IDE. Last but not least, one can find a jupyter notebook on the Github with an example of use which is easily reproducible. To make it so, the dataset CLS-french is provided with the library.

9.2 SCBert in a nutshell

Now that we have explained why we developed that tool let's dive into SCBert itself and what it does. In few words our tool is divided in 2 parts, generate unique vectors for each texts using FlauBERT model and in the second part cluster, extract keywords and compute a coherence measure.

9.2.1 The Vectorizer

The Vectorizer is the name we gave to the python class that generates a unique vectors for each texts. To generate those vectors there is few steps :

- **Load pre-trained Flaubert Model :** As discussed in the previous section, our dataset is in french so we decided to use french models. Hugging Face provides several pre-trained models and we allowed only 4 french pre-trained : Flaubert-small, Flaubert-base, Flauber-large and Camembert.
- **Tokenize :** Each pre-trained model comes with it's own tokenizer because models was trained in a special way with special cleaning method. Thus we load the tokenizer corresponding to the model chosen by the user at the previous step. Then the tokenizer loaded will split each texts in it's own way and also we will pad or cut each text to MAX LEN which is by default at 256 tokens but can be personalized. To keep in mind which token is padded or not we also generate an attention tensor with 1 and 0. The ones are at the same place as the real tokens, and for padded tokens there are 0s. Also for each token, the pre-trained model replace this word or part-of-word with a number corresponding to its index in the vocabulary. This step is necessary as like all algorithm words are not understandable by a computer.
- **Forward :** For that step we take as input the lists of ids and the attention tensors. Those two tensors are passed into the pre-trained model and as an output we store the value of every node for every layers of the models to be able personalized the way we generate word with pooling in the next step. Unfortunately, as all BERT models are big, the computational time is exponential. To give an idea, on our test set composed on 6000 reviews, doing the forward took approximatively 7 min for Flaubert small, 30min for Flaubert base and 2h for Flaubert large, on a computer with 16Go of RAM.
- **Pool :** This step of pooling is the main part of the Vectorizer and we can divide it into 2 parts :
  - **Word pooling :** is the way of generating a unique vector for each word. As said earlier we keep in memory all the weights of all layers of the forward pass. Then the user decide with which method to generate the unique word vectors. First option is to decide to take only the weight of one layer and then there is nothing else to do. But if they decide to use a combination of layers then 3 options are available : max, mean or concat. Max pooling for words just take the bigger value for each layer selected, the mean speak for itself and the concat just concat the weights of the layer selected thus for a 2 layers selected of size 768, each word will be represented by a tensor of size 1536.
  - **Sentence pooling :** After the word pooling we have a unique vector for every word of the texts, so to get a unique vector for each texts we have to do a similar task of pooling with the vector generated previously. Here we only make available the mean and max pooling.

A wrapper is also available which allows the user to call only one function to do all the previous steps.

9.2.2 The Embedding Explorer

The second class of our library is called EmbeddingExplorer because what we generated in the previous section was an embedding of every texts and this function will try to accomplish the purpose of our project which is : cluster the vectors and try to extract human interpretability. To extract that interpretability we will after clustering, extract keywords and then measure the coherence of the keywords that represents each clusters.

- **Cluster :** Our library wraps several clustering algorithm available with scikit learn such as k-means, MiniBatchKMeans, DBSCAN, agglomerative, SpectralClustering. Due to limitations in computation times, we mainly used MiniBatchKMeans because it is a faster implementation of kmeans. This function then return the list of the labels corresponding to the cluster output of a chose algorithm
- **Extract keywords :** To extract keywords within a given cluster we take only the texts with the same labels from the previous step and then we use python library called *multi-rake* which allows to generate keywords from a give texts (multi here stands for multi language but for now we only use the french language). As the function requires a unique text we concatenate all texts within a cluster to have a large one and then extract keywords of that text. The keywords are then stored in a variable to be used during the next step.
- **Compute C_v coherence :** The coherence C_v is then calculated on the keywords extracted for every clusters. A good cluster has a coherence superior to 0.5 and bad clusters have lower than 0.3 coherence. The higher this coherence, the easier for a human to understand the subject. To implement we used the CoherenceModel function of gensim which takes as input the keywords, the tokenized texts, and a dictionary of the word used in the tokenized texts.

A wrapper is available only for the 2 previous steps.

9.3 Architecture of the library

The source code of our tool follows the python package tutorial architecture as we wanted to follow some structured and well-known architecture. So the architecture of the Github is as follow :

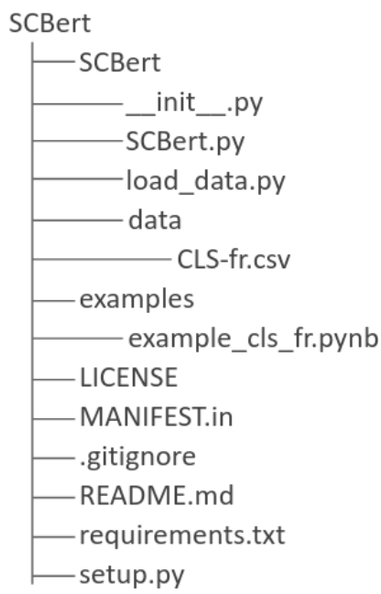


Figure 5: Architecture of the Github repository

As you can see on the previous figure the project's architecture is normal for a library as the file setup.py describes the title and everything else of the library, the requirements.txt are referencing the dependencies to other python libraries. The MANIFEST file is only here to say to include the cls dataset and the LICENSE just for the purpose of free rights.

On the graph above one can see how the library actually works. It is divided into 2 files :

- **load_data.py** : containing the class to load the CLS-fr dataset that comes within.
- **SCBert.py** : containing 2 classes, one for the Vectorizer and one for the EmbeddingExplorer as discussed earlier. Each class implements several functions that do the few steps described before.

10 Results Analysis

10.0.1 Our test dataset

The Cross-Lingual Sentiment (CLS) dataset comprises about 800.000 Amazon product reviews in the four languages English, German, French, and Japanese.

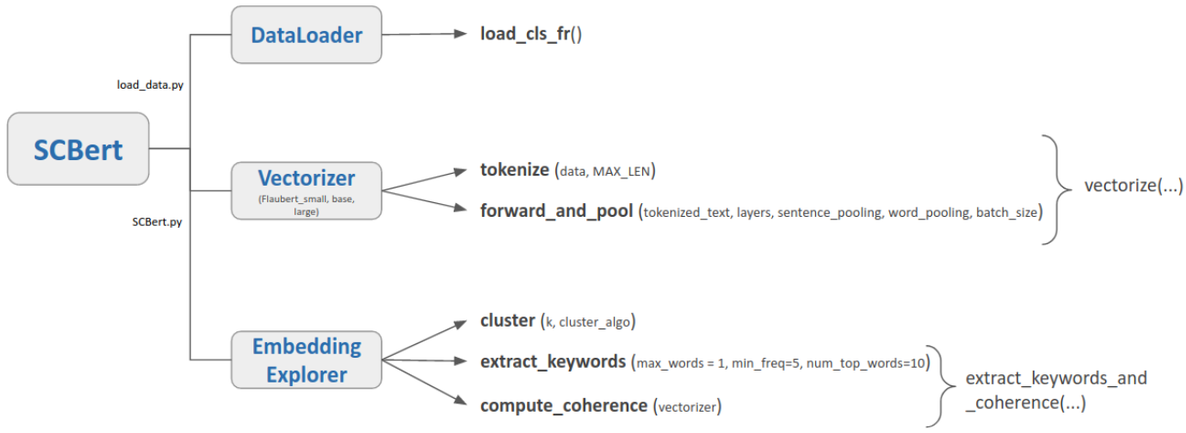


Figure 6: Architecture of the library and the functions works

The dataset was first used by (Prettenhofer and Stein, 2010). It consists of Amazon product reviews for three product categories—books, dvds and music—written in four different languages: English, German, French, and Japanese. The German, French, and Japanese reviews were crawled from Amazon in November, 2009. The English reviews were sampled from the Multi-Domain Sentiment Dataset (Blitzer et. al., 2007). For each language-category pair there exist three sets of training documents, test documents, and unlabeled documents. The training and test sets comprise 2.000 documents each, whereas the number of unlabeled documents varies from 9.000 - 170.000.

We are intrested in the french part only. In the research paper of FlauBERT, they use this data set to evaluate the performance of their algorithm of classification.

Each line corresponds a review and also the title of the article , the type of the article and the rating attributed by the users.

	category	title	review	rating	code
0	DVD	X-men origins - Wolverine - Edition simple	Alors que les premiers X-men avaient l'affecti...	2.0	0
1	DVD	X-men origins - Wolverine - Edition simple	on attendait bien plus sur cette génèse; les e...	2.0	0
2	DVD	Harry Potter et le prince de sang-mêlé - Editi...	Une histoire qui s'éloigne du livre, des scène...	2.0	0
3	DVD	X-men origins - Wolverine [Blu-ray]	Wolverine est de retour, sans les X-Men et por...	2.0	0
4	DVD	Kaamelott : Livre VI - Coffret 4 DVD	Grand fan de la période "épisodes courts et dr...	2.0	0
5	DVD	Star Trek , le film 2009 [Blu-ray]	VRAIMENT PAS CONTENT\n Le son n est pas du tou...	1.0	0
6	DVD	Le 10ème royaume - Coffret Collector 3 DVD	C'est un réel plaisir de replonger dans cette ...	4.0	0
7	DVD	LoL - Laughing out Loud	J'ai trouvé ce film très moyen. Un ersatz de "...	1.0	0
8	DVD	LoL - Laughing out Loud	ILS EN ONT FAIT DE LA PUBLICITE AUTOUR DE CE F...	1.0	0
9	DVD	Billy Elliot	Dans la lignée de "Full Monty", "My name is Jo...	5.0	0
10	DVD	James Bond : Quantum of Solace - Edition simple	Alors le voilà, le dernier James Bond. Autant ...	2.0	0

10.0.2 Probabilistic Topic Model

Latent Dirichlet Allocation (LDA) is a "generative probabilistic model" of a collection of composites made up of parts.

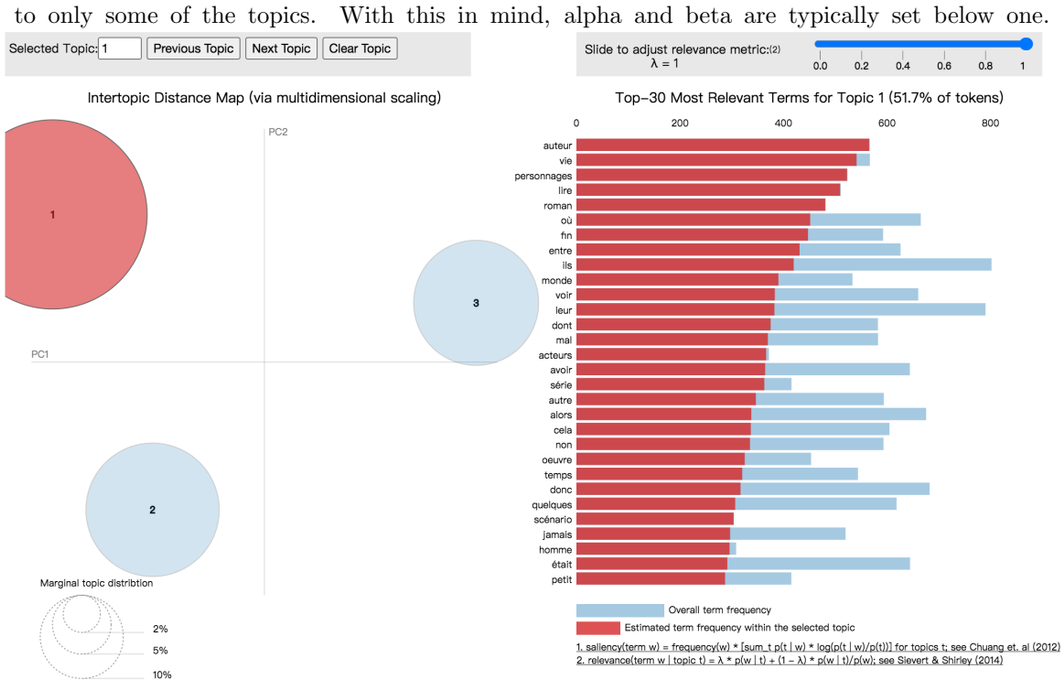
LDA is particularly suitable for our task because LDA is a way of soft-clustering composites and parts and also because we would view the number of topics as a number of clusters and the probabilities as the proportion of cluster membership.

Contrast this with say, k-means, where each entity can only belong to one cluster (hard-clustering). LDA allows for 'fuzzy' memberships. This provides a more nuanced way of recommending similar items, https://www.overleaf.com/project/5e3305995e63e000015e9a13finding duplicates, or discovering user profiles/personas.

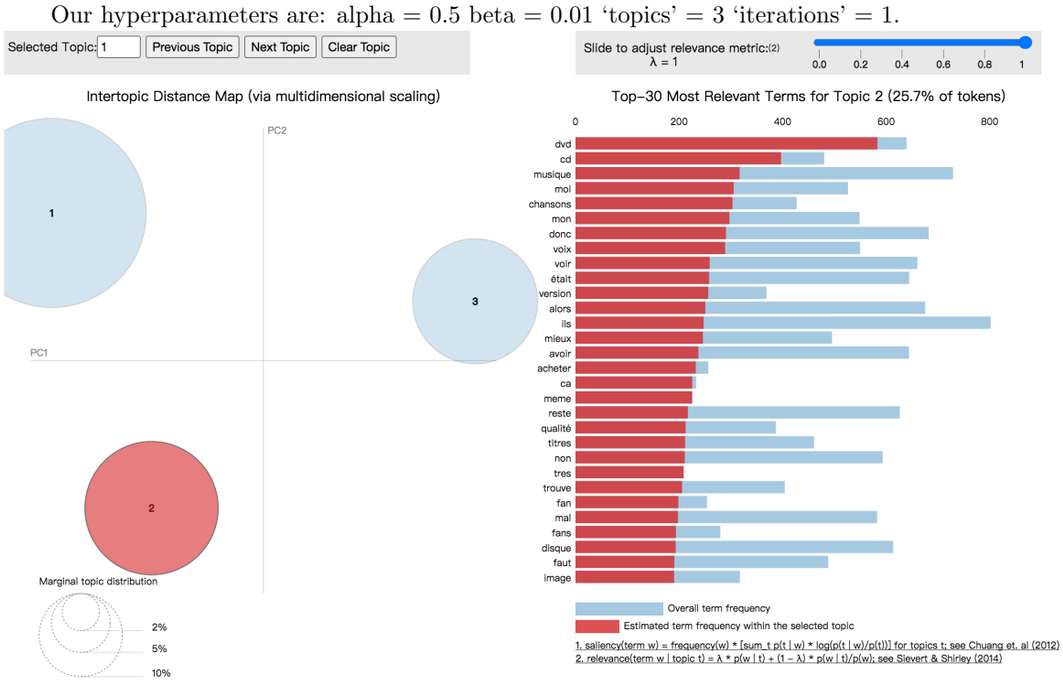
There are two hyperparameters are required by LDA. The alpha controls the mixture of topics for any given document. Turn it down, and the documents will likely have less of a mixture of topics. Turn it up, and the documents will likely have more of a mixture of topics.

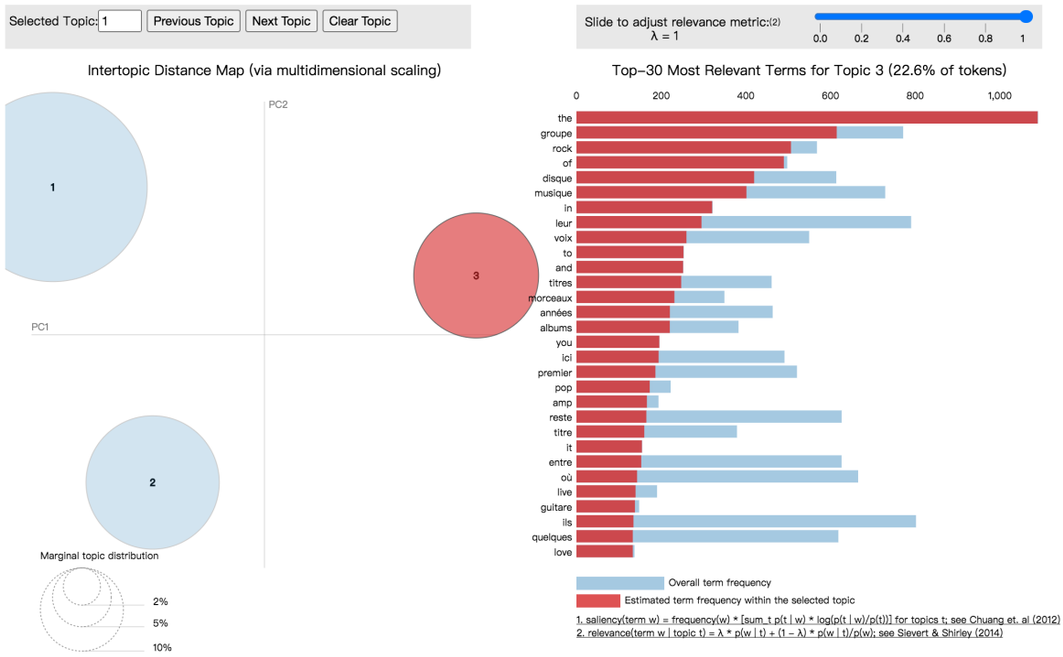
The beta hyper parameter controls the distribution of words per topic. Turn it down, and the topics will likely have less words. Turn it up, and the topics will likely have more words.

Ideally, we want our composites to be made up of only a few topics and our parts to belong to only some of the topics. With this in mind, alpha and beta are typically set below one.



Our hyperparameters are: alpha = 0.5 beta = 0.01 'topics' = 3 'iterations' = 1.





We can see the topics keys words found here have lots of senses.

Then we measure the coherence of the key words of each cluster using C-v coherence as our comparison metric.

	Keywords	**C_v Coherence:**
Cluster 1	**auteur, roman, vie, lira, personnages,monde, page, fin, homme, temp, amour, ouvrage, mal, lecture,lu**	**0.5220120616179623**
Cluster 2	**groupe, rock, disque, morceau, cd, voix, chansons, musique, fan, écouter, écoute, pop, you, ca**	**0.5999509514284544**
Cluster 3	**dvd, version, image, voir, musique, acteurs, scène, qualité, qualité, série, scénario, mal, fan, oeuvre, mieux, années**	**0.36256458135377795**

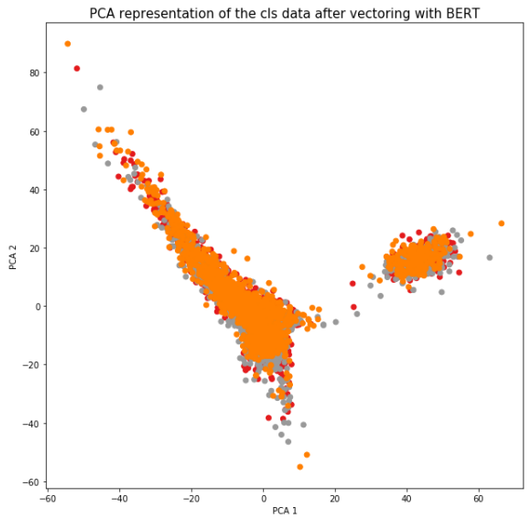
10.0.3 Word Embedding Model

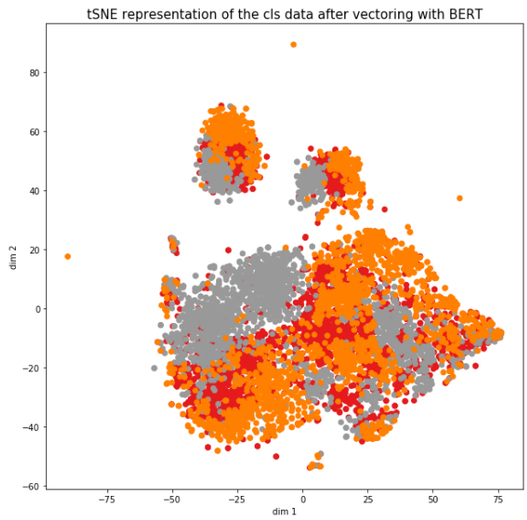
We use the embedding of each word in the text than we do a average of all the word embeddings in a text to get a vectorial representation of each text. Since this representation is a dense vector, it's easier to apply a clustering method based on distance metric like k means.

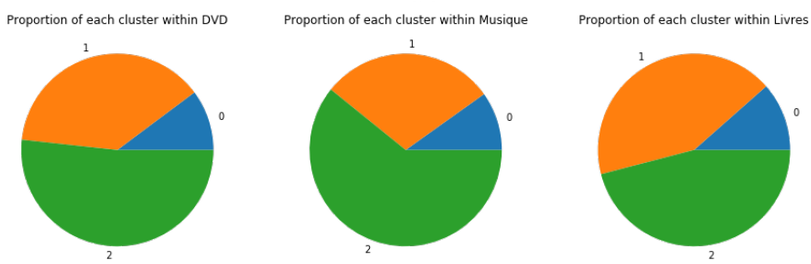
Our hyperparameters are:

word_pooling_method	max,**average**
sentence_pooling_method	max,**average**,concat
layers	**20,** [19,20,21]
MAX_LEN	256
batch_size	50
model_name	**flaubert_large**
Clustering	**KMeans**,MiniBatchKMeans, AgglomerativeClustering, DBSCAN
RAKE	max_words =1, min_freq=5, num_top_words=15

Here is a 2D PCA and tSNE representation of k means clustering with FlauBERTlarge model.







The color in the pie chart corresponds the clusters. We can see in each cluster, the proportion of DVD/Music/Book categories varies.

After getting the clusters, then we use RAKE (rapid automatic keyword extraction) to get the key words of each cluster. Then we measure the coherence of the key words of each cluster using C-v coherence as our comparison metric.

The results is encouraging but the keyword extraction method is still to be improved in order to represent better the clusters.

10.0.4 Comparison

	Keywords (FlauBert)	**C_v Coherence:**	Keywords**(LDA)**	C_v Coherence:
Cluster 1	c'est, demander, plus, bonus, dommage, bonheur, malgré, intéressant, famille, monde, place, film' "n'ai" 'réussi, pourtant	**0.33**332199645532984	auteur, roman, vie, lira, personnages,monde, page, fin, homme, temp, amour, ouvrage, mal, lecture,lu	**0.52**20120616179623
Cluster 2	mal, cerveau, scènes, point, idéal, restera, service, film, nombreux' "l'histoire" 'vie, générique, public, sensible,n'y	**0.37**31836736270943	groupe, rock, disque, morceau, cd, voix, chansons, musique, fan, écouter, écoute, pop, you, ca	**0.59**9950951428454
Cluster 3	retour, porté, toujours, grâce, pouvoir, jour, propose, rejoindre, plutôt, subir, trahison, sent, place,l'aise,quitte	**0.35**86242745099887	dvd, version, image, voir, musique, acteurs, scène, qualité, qualité, série, scénario, mal, fan, oeuvre, mieux, années	**0.36**2564581353777

For our final conclusion, we would focus on C-v coherence measurement.

In terms of coherence measurement, the LDA model perform better. There are several reason for this:

- we didn't explore all the hyper-parameter space of FlauBERT,we can only use the pre-trained model without doing fine-tuning, the text embedding can be improved;

- the Latent Dirichilet Allocation extract the salient words directly based on the latent distributions; the text embedding model take into account the embedding of each word in each text, these are completely different methods.

- we use RAKE to extract the key words of each cluster, this word extraction approach can be improved.

11 Conclusion

Despite the difficulty of working on IBM's final dataset for confidentiality reasons, we tried to take into account the constraints of IBM's customers, so we took a French dataset to build our models and measure the coherence of each cluster. After few months of research into unsupervised Natural Language Processing tasks and how to apply those different approaches to our problem we decided to go further with 2 different approaches. The first one is the Latent Dirichlet Allocation (LDA). It is a statistical method on which we plugged a measure of coherence to estimate the human interpretability. The second one is the famous BERT algorithm which uses the auto-attention mechanism. We chose that method along with IBM as we all wanted to explore that option because it is currently state-of-the-art in numerous tasks of Natural Language Processing. As this method is quite hard to play with we decided to provide a tool named SCBert to IBM. On our test data, unfortunately the LDA method outperformed our tool SCBert. However, one have to keep in mind that the space of hyper-parameter of SCBert is close to infinite and exploring that space would require an infinite amount of time thus we cannot be sure that it is a bad option. We want to end this project by thanking IBM for their close support and the bi-mensual meeting we had. We also want to thank our academic tutor Chloe Clavel.