الجمهوريّة الجزائريّة الديمقراطيّة الشعبيّة
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of High Education and Scientific Research
جامعة غرداية
University of Ghardaia
كليّة العلوم والتكنولوجيا
Faculty of Science and Technology
قسم الرياضيات والإعلام الآلي
Mathematics and Computer Science Department

Registration N°:
/…../…../…../…../…..

Thesis

For obtaining the master's degree

Domain: Artificial Intelligence and Linguistics
Field: Computer Science
Specialty: Intelligent Systems for Knowledge Extraction

Theme

Topic Modeling with Word Embeddings

Submitted on 29/09/2021

By
Ahmed ITBIRENE and Brahim CHIHANI

Examined by the jury composed of:

Dr. Slimane OULAD-NAOUI	MCB	University of Ghardaia	Examiner
Dr. Houssem Eddine DEGHA	MAB	University of Ghardaia	Examiner
Dr. Slimane BELLAOUAR	MCA	University of Ghardaia	Advisor

Academic Year: 2020/2021

ملخص

مع التطور الكبير في مجال الرقمنة فإن استخراج المواضيع من خلال المعلومات التي تكون في أغلب الآحيان على شكل نصوص غير موسومة ليس بالأمر الهين, لذلك فإننا نحتاج إلى تقنية نمذجة المواضيع (Topic Modeling) **والتي بدورها تستند إلى خوارزميات غير خاضعة للإشراف.**

من خلال مذكرتنا نقوم بتوضيح مفهوم نمذجة المواضيع والمناهج المرتبطة بها, نذكر من بينها:

Latent Dirichlet Allocation (LDA), Embedded Topic Model (ETM), Gaussian LDA (G-LDA), LDA with Word2Vec (LDA2Vec).

من ناحية العمل التجريبي، نقوم بإجراء مقارنة بين الطريقتين (LDA) و (ETM) على مجموعة 20 newsgroups.

حيث كانت النتائج التي تحصلنا عليها من حيث وقت التنفيذ وتناسق المواضيع لصالح طريقة (ETM).

الكلمات المفتاحية: نمذجة المواضيع, تناسق المواضيع, (LDA) Latent Dirichlet Allocation, (ETM) Embedded Topic Model, (G-LDA) Gaussian LDA, (LDA2Vec) LDA with Word2Vec.

Abstract

With the great development in the field of digitization, the extraction of topics through information that is in the form of unmarked texts, is not an easy matter. Therefore, we need a topic modeling technique, which is based on unsupervised algorithms.

In our thesis, we clarify the concept of topic modeling and the inherent approaches, such as Latent Dirichlet Allocation (LDA), Embedded Topic Model (ETM), Gaussian LDA (G-LDA), and LDA with Word2Vec (LDA2Vec).

In the experimental work, we make an empirical comparison between both LDA and ETM methods on the 20 newsgroups, in terms of runtime and topic coherence. The results are in favor of the ETM method.

Keywords: topic modeling, topic coherence, Latent Dirichlet Allocation (LDA), Embedded Topic Model (ETM), Gaussian LDA (G-LDA), LDA2Vec.

Résumé

Avec le grand développement dans le domaine de la numérisation, l'extraction des sujets à travers des informations qui se présentent sous forme de textes non étiqueté, n'est pas une tache facile. Par conséquent, nous avons besoin d'une technique de modélisation thématique basée sur des algorithmes non supervisés.

Dans notre thèse, nous clarifions le concept de modélisation thématique et les approches associées tel que Latent Dirichlet Allocation (LDA), Embedded Topic Model (ETM), Gaussian LDA (G-LDA), et LDA avec Word2Vec (LDA2Vec).

Dans le travail expérimental, nous faisons une comparaison empirique entre les deux méthodes LDA et ETM, sur le 20 newsgroups, en termes de temps d'exécution et de cohérence thématique,

Les résultats que nous avons obtenus sont en faveur de la méthode ETM.

Mots clés: modélisation thématique, cohérence thématique, Latent Dirichlet Allocation (LDA), Embedded Topic Model (ETM), Gaussian LDA (G-LDA), LDA2Vec.

Dedications

First of all, I thank the Almighty God for the guidance, strength, power of the mind, protection, and skills, and for healthy living.

This thesis is wholeheartedly dedicated to my beloved parents, who have been my source of inspiration and gave me strength in my weaknesses, who continually provide their moral, spiritual, emotional, and financial support,

To my amazing wife, whose sacrificial care for me made it possible for me to complete this work,

To my dear grandparents and my mother and father in law,

To my brothers, sisters, relatives, mentor, friends, and classmates who shared their words of advice and encouragement to finish this study,

I hope that this achievement will complete the dream that you had for me all those many years ago.

Ahmed

Dedications

I dedicate this modest work to the soul of my dear mother, may God have mercy on her in his vast paradise.

My deepest feelings for my dear father, i wish him a speedy recovery.

To my very dear wife.

To my beautiful children Mohammad, Hammu and Ali.

To all my brothers and sisters and family.

To all my friends and everyone who loves me.

To all my colleagues at work, especially the office of Computer Science of the municipality of Ghardaia.

Brahim

Acknowledgment

We thank God first and foremost for giving us the courage and health to accomplish this Modest work.

This work could not have led to results without the help and encouragement of the many people we thank.

We thank our supervisor Dr. Slimane BELLAOUAR, for his help, understanding, advice, criticism and encouragement. We would like to thank the members of the jury for the honor they have given us by accepting to judge our work.

We also thank our co-workers for their encouragement and support. Finally, we wish everyone who helped and encouraged us from near or far in the realization of this work to find here our gratitude and sincere thanks.

Contents

List of Tables

List of Figures

List of Algorithms

List of Listings

Introduction

Since the explosion of information with the advent of the third generation of web sites, classic data analysis approaches have become helpless in front of the massive flow of information. Then it is necessary to find more effective and smarter ways to confront this phenomenon, especially when the information is unlabeled text type. But is not easy to classify this information in a specific topics, thus it is advantageous to make reference to unsupervised algorithms, namely topic modeling.

Topic modeling is a field of text mining that uses unsupervised machine learning techniques for extracting the meanings of words according to their context. It can take huge collection of documents, grouping words into bag of words, identifying topics, by using a similarity process.

In our work, we present different approaches of topic modeling starting with the classical method named Latent Dirichlet Allocation (LDA) (Blei et al. (2003)), and passing through several new methods that are devised from the combination of LDA model and Word Embedding technique. Those methods are Embedded Topic Model (ETM) (Dieng et al. (2020)), Gaussian LDA (G-LDA) (Das et al. (2015)), and finally LDA with Word2Vec (LDA2Vec) (Moody (2016)).

As a contribution to this thesis, we made an experimental comparison between both LDA and ETM methods, using the corpus of 20Newsgroups. The results we obtained in terms of runtime and topic coherence, are in favor of the ETM method.

We divided this thesis into three chapters. In the first chapter we present some basic concepts and definitions about Natural Language Processing (NLP), Machine Learning (ML), and then some detail of topic modeling with touching on word embedding. In the second chapter we present a detailed study of the different approaches of topic modeling. In the last chapter we make an experimental comparison between LDA and ETM methods. We choose as a corpus the 20 newsgroups and as a word embedding model the pre-trained Google's Word2Vec. Finally, the conclusion summarizes what we passed through in our thesis.

Chapter 1

1 Background

In this chapter, we present the necessary information for understanding the continuation of this work. We base first on Machine Learning (ML), then Natural Language Processing (NLP), and then text pre-processing, and finally text representation with word vectors and NLP modeling, which are used by different approaches of topic modeling.

1.1 Machine Learning

Machine Learning (ML) is a field of Artificial Intelligence (AI) that relies on mathematical and statistical approaches to give computers the ability to learn from data, improving their performance in solving tasks without being programmed (Jordan and Mitchell (2015)).

1.1.1 Supervised Learning

In supervised learning, we train the machine using data that is well labeled. It means supervised learning algorithms learns from labeled training data, and help us to predict outcomes for unforeseen data. Supervised machine learning is split into two main branches:

- **Classification (categorical)**

Classification means grouping the output into a class (or multiple classes). If the algorithm tries to label the input in two separate classes, this is called a binary classification. The selection between more than two classes is called a multiclass classification.

- **Regression (quantitative)**

The regression technique uses the training data to predict a single continuous output value.

1.1.2 Unsupervised Learning

Unsupervised learning is a machine learning technique in which the model does not need supervision. We let the model to work on its own to discover information. It mostly deals with data that has not been labled.

- **Clustering**

Clustering is a field of unsupervised learning. It can find a structure or pattern in a collection of uncategorized data.

- **Association (rules)**

Association rules allow us to establish associations amongst data objects inside large databases. This unsupervised technique is about discovering exciting relationships between variables in large databases.

Figure 1.1 Describes the Machine learning structure.

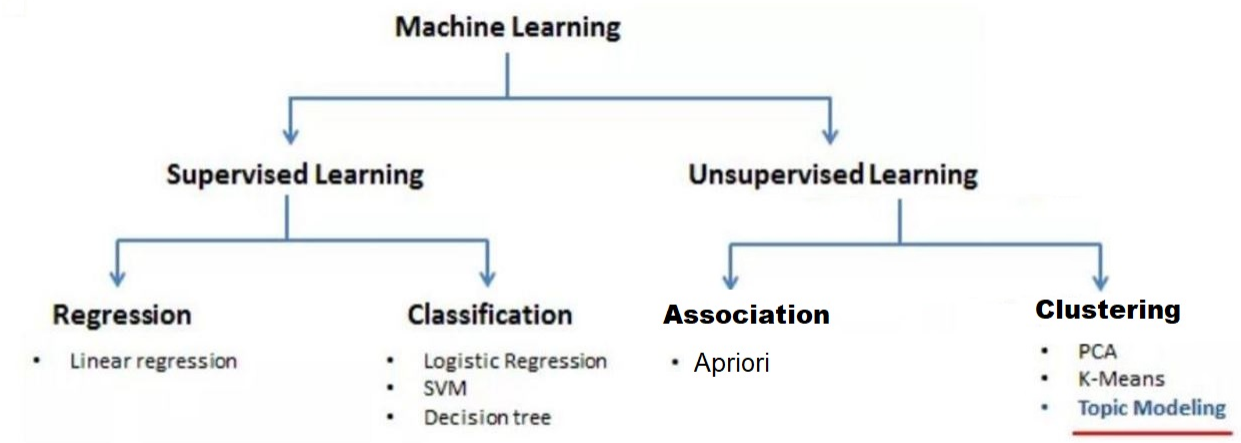


Figure 1.1: Machine learning structure.

1.2 Natural Language Processing (NLP)

Natural Language Processing or NLP is the sub-field of machine learning that is focused on enabling computers to understand and process human language (Vijayarani et al. (2015)). In other words, with NLP computers are taught to understand human language, their meanings and sentiments.

Some applications of natural language processing are:

- Chatbots.
- Speech Recognition.
- Audio Processing.
- Text Classification.
- Topic Modeling.

Topic Modeling

Topic modeling is an NLP approach that is used for automatically discover the abstract topics that occur in a collection of documents (corpus of text data).

This process is an unsupervised technique similar to clustering on numeric data, which means that we do not have to provide a labeled dataset to topic modeling algorithms, and topics are identified automatically by the model.

A document can be part of multiple topics (Rani and Kumar (2021)), like in fuzzy clustering (soft clustering) in which each instance (document) can belong to each cluster (topic) with a certain degree. Figure 1.2 explains the working principle of topic modeling.

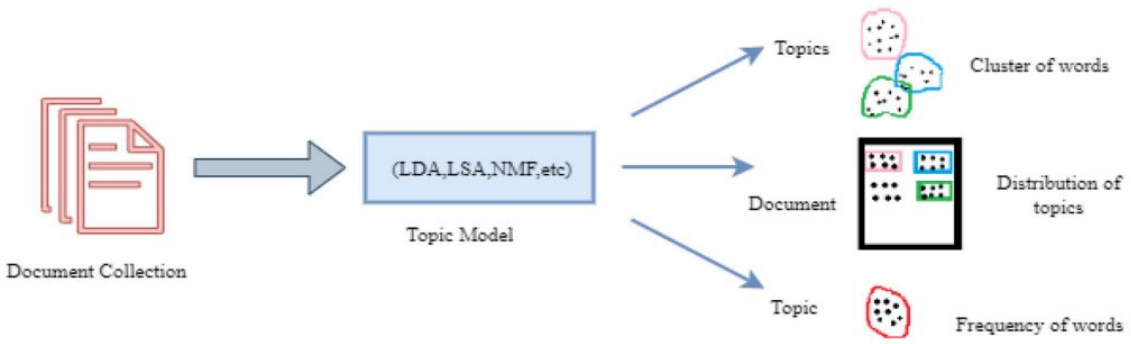


Figure 1.2: Topic modeling diagram (Rani and Kumar (2021)).

Topic modeling provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives.

It can help with the following:

- Discovering the hidden themes in the collection,
- Classifying the documents into the discovered themes,
- Using the classification to organize/summarize/search the documents.

Let us say a document belongs to the topics (food, dogs and health). So if a user queries "dog food", they might find the above-mentioned document relevant because it covers those topics (among other topics). We are able to figure its relevance with respect to the query without even going through the entire document. Therefore, by annotating the document, based on the topics predicted by the modeling method, we are able to optimize our search process.

1.3 Text Pre-Processing

Text preprocessing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form (Vijayarani et al. (2015)) so that machine learning algorithms can perform better as shown in Figure 1.3.

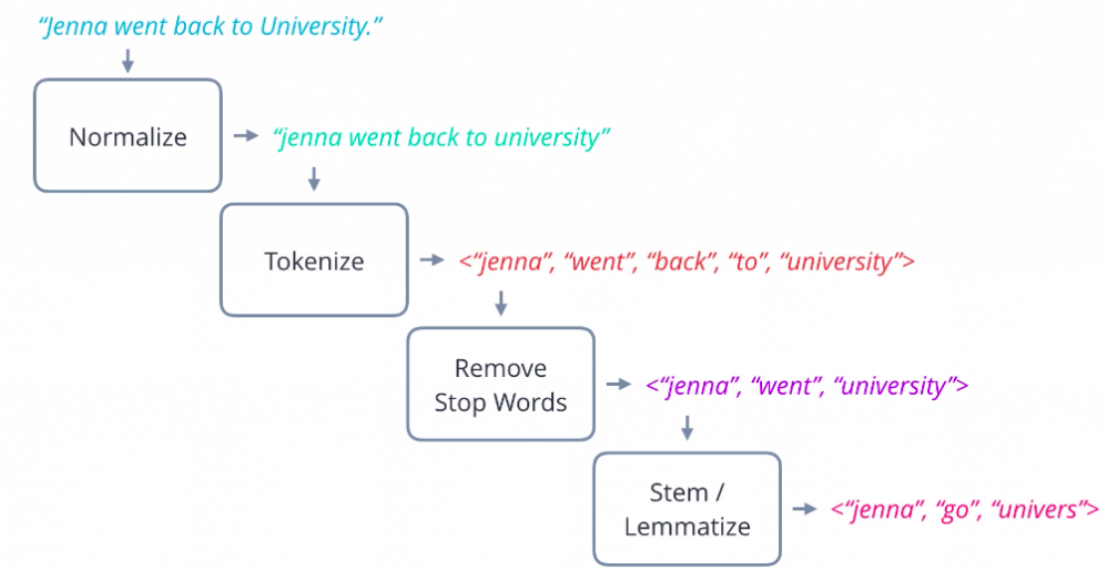


Figure 1.3: Text pre-processing techniques.

Normalization

Normalization is the process of lowercasing and cleaning the corpus from non-alphabetic characters (e.g., numbers, punctuations, and special characters).

Tokenization

Tokenization is the process of splitting documents into units of observations. We usually represent the tokens as n-gram; where n represent the consecutive words occurring in a document. In the case of unigram (one word token), the sentence "David works here" will be tokenized into: "David", "works", "here". In the case of bigram (two words token), the sentence "David works here" will be tokenized into: "David works", "works here".

Stopwords Removal

Stopwords removal is the process of removing all words that are semantically meaningless in the corpus. A list of stopwords should be provided.

Lexicon Normalization

Aside from stopwords, a different type of noise can arise in NLP. For example, collect, collection, collected, and collecting are all similar words. Using stemming and lemmatization would reduce all variations of the same word (noun, verb, adverb, adjective, etc.) to a common base form.

- **Stemming**

Stemming allows us to remove different variations of the same word. For example, change, changes, changed and changing will all be reduced to the same single word "chang". Figure 1.4 illustrates it.

- Stemming is the process of reducing inflection in words to their root form, such as mapping a group of words to the same stem even if the stem itself is not a valid word in the language.
- Stemming often leads to incorrect meanings and spelling.
- Stems are created by removing the suffixes or prefixes used with a word.

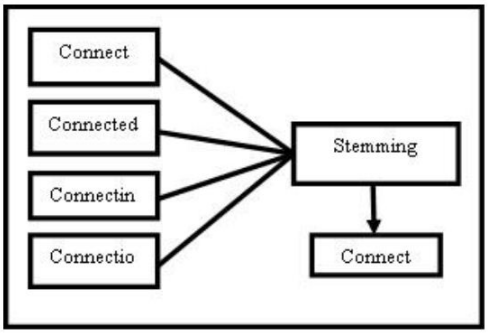


Figure 1.4: Stemming example (Vijayarani et al. (2015)).

- **Lemmatization**

The only difference between lemmatization and stemming is that lemmatization returns meaningful base form.

For example, instead of returning "chang" like stemming would. "change" will be returned.

- Unlike stemming, lemmatization reduces the inflected words properly ensuring that the root word belongs to the language.
- In lemmatization, the root word is called "lemma".
- A lemma (plural lemmas) is the canonical form, dictionary form, or citation form of a set of words, we note that stemmers and lemmatizers work on individual words so we have to tokenize the data first so that we can apply stemming or lemmatization to it.

1.4 Word Vectors and NLP Modeling

Computers are unable to understand the concepts of words. In order to process natural language, a mechanism for representing text is required.

The standard mechanism for text representation are word vectors where words or documents from a given language vocabulary are mapped into vectors of real numbers.

1.4.1 Traditional Word Vectors

Before diving directly into Word2Vec it is worth while to do a brief overview of some of the traditional methods that pre-date word embeddings.

Bag-Of-Words (BOW)

Bag-Of-Words (BOW) model (Kim et al. (2017)) is a text representation that describes the occurrence of words within a document. It involves two elements:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called "bag of words" because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document. This process is often referred to as vectorization.

Let us understand this concept with an example. Suppose we want to vectorize the following:
Doc1: "I am great".
Doc2: "Tom is very lazy".
Doc3: "George is good and active person".

For this small example, let us treat each line as a separate "document" and the three lines as our entire corpus of documents.

Step 1: Determine the vocabulary: We first define our vocabulary, which is the set of all words found in our document set. The only words that are found in the three documents above are: (I, Tom, George, am, is, great, good, lazy, active, very, and, person).

Step 2: Counting to vectorize our documents. All we have to do is counting how many times each word appears as shown in Table 1.1.

Documents	I	Tom	George	am	Is	great	Good	lazy	active	Very	and	person
Doc1	1	0	0	1	0	1	0	0	0	0	0	0
Doc2	0	1	0	0	1	0	0	1	0	1	0	0
Doc3	0	0	1	0	1	0	1	0	1	0	1	1

Table 1.1: Document-term matrix.

The bag-of-words model is very simple to understand and implement and offers a lot of flexibility for customization on your specific text data. It has been used with great success on prediction problems like language modeling and document classification. Nevertheless, it suffers from some shortcomings (Kim et al. (2017)), such as:

- **Sparsity:** Sparse representations are harder to model both for computational reasons (space and time complexity) and for information reasons, where the challenge is for the model to harness so little information in such a large representational space.
- **Meaning:** Discarding the word order ignores the context and consequently the meaning of words inside documents (loss of semantics and contextual information). Context and meaning can offer a lot to the model, that if modeled could detect synonyms ("good" vs. "great"), antonyms ("active" vs. "lazy"), and much more.

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF (Ramos et al. (2003)) is a text representation that is intended to reflect how important a word is to a given document inside a collection (corpus). It provides some weighting to a given word based on its occurrence inside documents. The TF-IDF value increases proportionally to the number of times the word appears in a document, but is offset by the number of documents in the corpus that contain that word, which helps to adjust the fact that some words appear more frequently than others.

- **TF (Term Frequency)**

TF is simply the frequency of words in a document, and it can be represented as the number of times a term shows up in a document.

- **IDF (Inverse Document Frequency)**

IDF represents the measure of how much information the word provides, i.e., if it is common or rare across all documents.

$$idf(w) = log(\frac{number\ of\ docs}{number\ of\ docs\ containing\ w}).$$

TF-IDF is the product of term frequency and inverse document frequency, or **TF*TDF**.

The TF-IDF formula:

$$W_{i,j} = tf_{i,j} \times log(\frac{N}{df_i}).$$

where:

- $tf_{i,j}$: number of occurrence of term i inside document j.
- df_i: number of documents containing the term i.
- N: total number of documents.

However, even though BOW and TF-IDF representations provide weights to different words they are unable to capture the word meaning.

1.4.2 Word Embeddings

Word embedding is trying to teach computers or machines the meaning of words. Recently, the NLP field has developed new linguistic models that rely on a neural network architecture instead of more traditional n-gram models. These new techniques are a set of language modeling and feature learning techniques where words are transformed into vectors of real numbers; hence, they are called word embeddings.

Word embedding models map all the words in a given language into numerical vectors of a given dimension. These models have quickly become popular because, once we have real numbers instead of strings, we can perform calculations. For example, we can find words that have the same context (synonyms) by calculating the distance between vectors.

Word2Vec

Word2Vec is the standard method for training word embeddings, which gained its popularity since 2013 (Mikolov et al. (2013)). It is the first method that demonstrated vector arithmetics to solve word analogies (Allen and Hospedales (2019); Ethayarajh et al. (2019)). Figure 1.5 illustrates some examples of the concept.

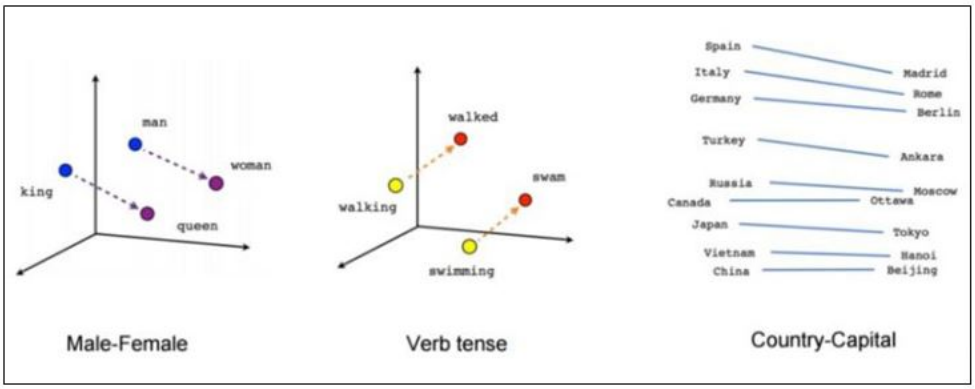


Figure 1.5: Linear word analogies.

Word2Vec tries to make the words with similar contexts have similar embeddings. In other words, instead of mapping the meaning of words (like dictionaries) we can rather infer the meaning of words that frequently appear around it. Therefore, when a word appears in a document its context is the set of words that appear nearby usually within a fixed size window. Let us consider an example to understand the concept better.

Example

- The kid said he would grow up to become a man.

- The child said he would grow up to become a man.

We can observe that the words kid and child are two different words with same context, and since Word2Vec makes words with similar context have similar embeddings then kid and child will have similar vectors when we iterate through a large corpus we would get a lot of sentences where kid is replaceable by child and hence these vectors will have similar embeddings.

Word2Vec converts word into vectors and with those vectors, we can have multiple operations like add, subtract and calculate distance between vectors to perform linear word analogies, and that is how the relationship among the words are preserved. Therefore, one example of this relationship is a very famous result of Word2Vec, which says:

$$\vec{king} - \vec{man} + \vec{woman} = \vec{queen}.$$

This form is a parallelogram structure of the linear analogy in the vector space (Figure 1.6).

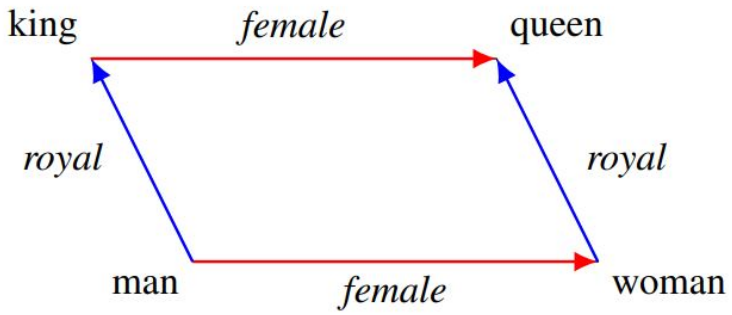


Figure 1.6: The parallelogram structure of the linear analogy (king,queen)::(man,woman). (Ethayarajh et al. (2019))

Definitions

- One-hot vector means one bit is set to 1 and all others are set to 0.
- Softmax is a non-linear function (Figure 1.7) that is used in neural networks as an activation function. It works in multi-class classification problems. The values outputted by nodes in a softmax layer will always sum to 1. When we are performing classification, these values are directly interpretable as probabilities.

The mathematical formula corresponding to the softmax function is given as follows:

$$f(x_i) = \frac{e^{(x_i)}}{\sum_{c=1}^{k} e^{(x_c)}}.$$

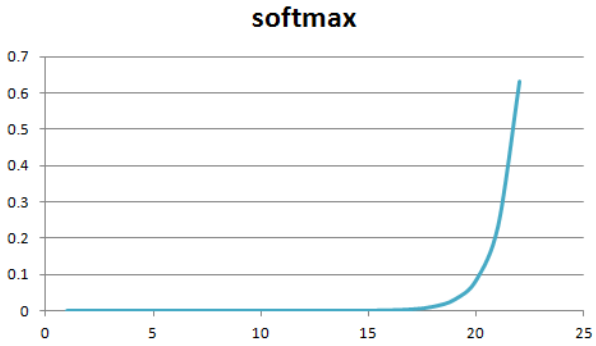


Figure 1.7: Softmax function graph.

Word2Vec has proven to be successful on a variety of downstream natural language processing tasks. It provides two main architectures that are used to produce a distributed representation of words; Continuous Bag-Of-Words (CBOW) and Skip-Gram (SG) (Figure 1.8).

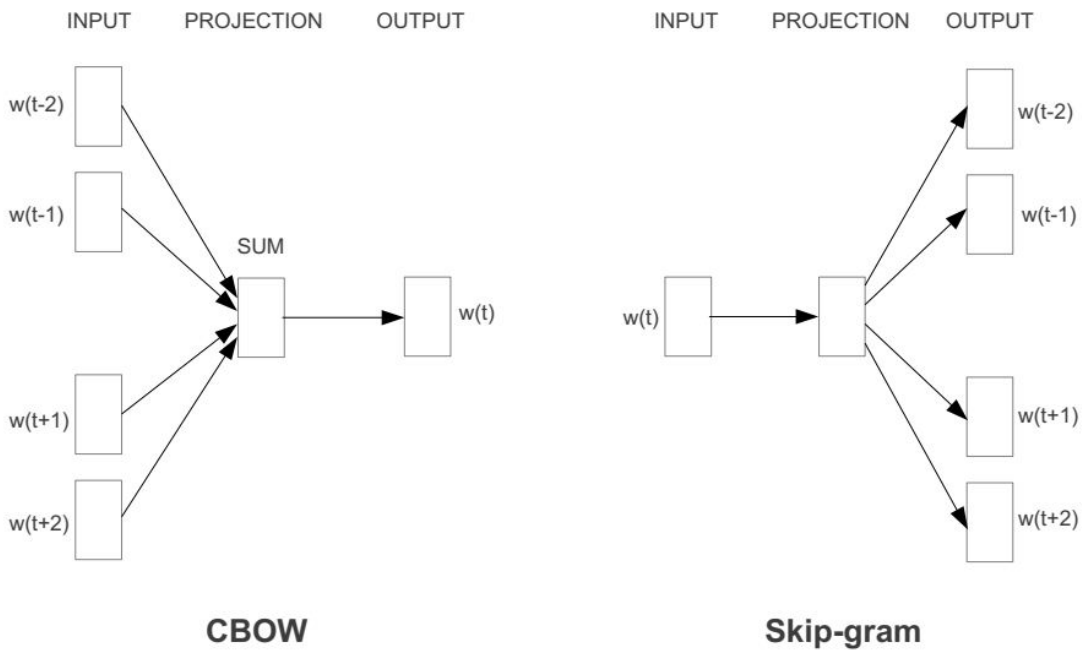


Figure 1.8: Architecture of Word2Vec models: CBOW and Skip-Gram. (Mikolov et al. (2013))

Notes

- The only activation function that is used in Word2Vec neural network models is Softmax function, and there is no other activation functions used like Sigmoid, Tanh or ReLU.
- Softmax is used in Word2Vec models in the output layer for calculating probability distributions.

Now, let us take a look at some more details about Word2Vec models.

- **Continuous Bag-Of-Words (CBOW)**

The model uses a context to predict target word. The context consists of a few words before and after the current word (Figure 1.9).

Example:

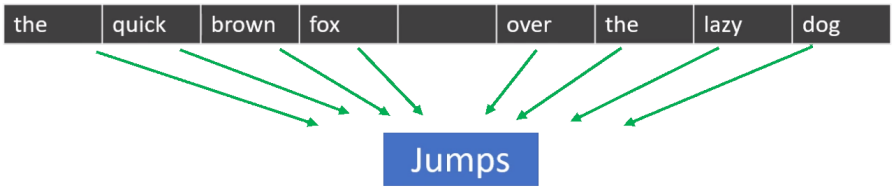


Figure 1.9: Continuous bag-of-words example.

The architecture of continuous bag-of-words neural network model is given in Figure 1.10:

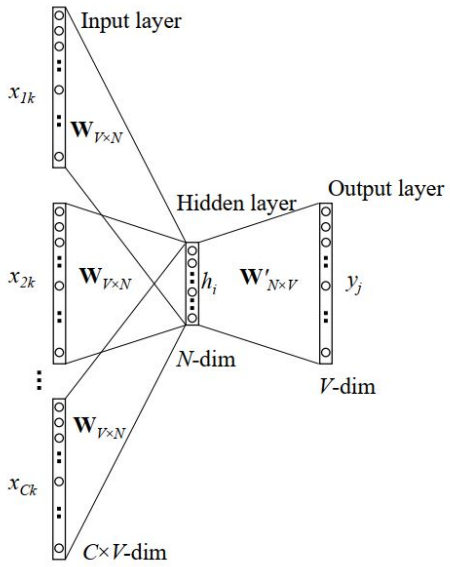


Figure 1.10: Continuous bag-of-words architecture.
(Rong (2014))

- The input layer has C one-hot encoded vectors of size V for the context words.
- The hidden layer contains N neurons (the size of the embedded vector).
- The output layer is a V length vector that contains the predicted probability distribution for the target word calculated by softmax activation function.
- W_{vn} is the weight matrix that maps the input x to the hidden layer ($V \times N$ dimensional matrix).
- $W'nv$ is the weight matrix that maps the hidden layer outputs to the final output layer ($N \times V$ dimensional matrix).

We Note that the input weights matrix W is shared by all the context words.

- **Skip-Gram (SG)**

The model uses a word to predict target context, i.e., predict words within a certain range before and after the current word (Figure 1.11).

Example:

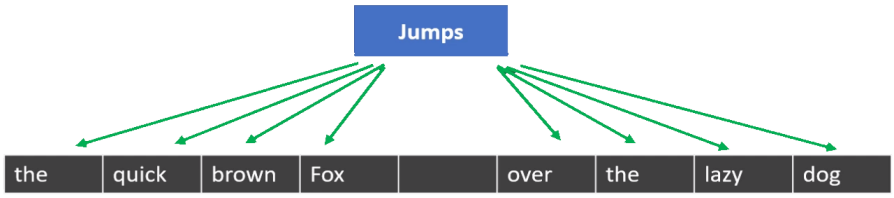


Figure 1.11: Skip-gram example.

The architecture of skip-gram neural network model is depicted Figure 1.12.

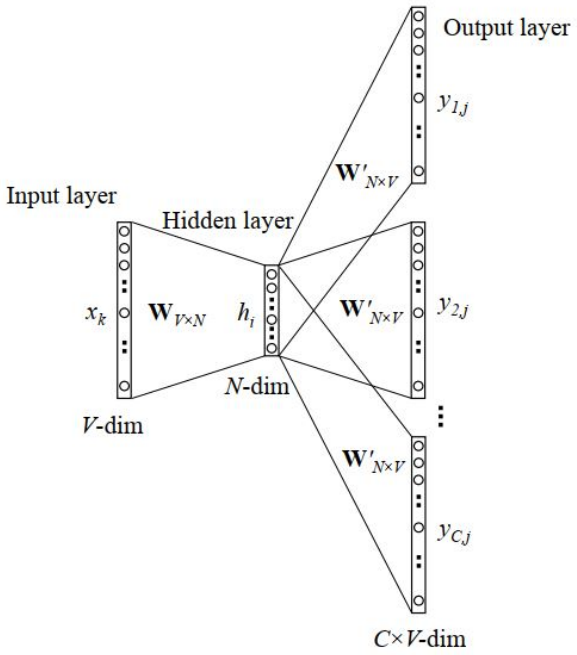


Figure 1.12: Skip-gram architecture.
(Rong (2014))

- The input layer is the one-hot encoded vector of size V for the middle word.
- The hidden layer contains N neurons (the size of the embedded vector).
- The output layer has C vectors of length V that contains the predicted probability distributions for each word from the target context words calculated by softmax activation function.
- W_{vn} is the weight matrix that maps the input x to the hidden layer ($V \times N$ dimensional matrix).
- W'_{nv} is the weight matrix that maps the hidden layer outputs to the final output layer ($V \times N$ dimensional matrix).

We note that the output weights matrix W' is shared by all the context words.

1.5 Topic Coherence Measures

Topic coherence measures are used for evaluating topic models to extract the best in terms of semantic similarity of words in a topic. We introduce four metrics:

1. The UCI coherence is an extrinsic measure introduced by Newman et al. (2010). It uses the Pointwise Mutual Information (PMI) as a pairwise score function of all the n-top words pairs. The UCI is calculated as follows:

$$UCI(w_i, w_j) = log\frac{P(w_i, w_j) + 1}{P(w_i).P(w_j)}.$$

- One was added to avoid the logarithm of zero.

where:

- $P(w_i)$: the probability of seeing the term w_i in a random document.
- $P(w_i, w_j)$: the probability of seeing both terms w_i and w_j co-occurring in a random document.

2. The $NPMI$ coherence measure is an enhanced version of the UCI coherence using the Normalized Pointwise Mutual Information (NPMI) (Aletras and Stevenson (2013)). It is calculated as follows:

$$NPMI(w_i, w_j) = \frac{log\frac{P(w_i, w_j+1)}{P(w_i).P(w_j)}}{-log(P(w_i, w_j) + 1)}.$$

3. The C_v coherence measure combines the indirect cosine measure with the Normalized Pointwise Mutual Information (NPMI) and the boolean sliding window (Röder et al. (2015)).

4. The $UMass$ coherence measure is an intrinsic measure based on document co-occurrence. It is calculated as follows:

$$UM_{ass}(w_i, w_j) = log\frac{D(w_i, w_j) + 1}{D(w_i)}.$$

where:

- $D(w_i)$: the number of documents containing the term w_i.
- $D(w_i, w_j)$: the number of documents containing both terms w_i and w_j.

Chapter 2

2 Related Work

In this chapter, we discuss different topic modeling approaches. We start with Latent Dirichlet Allocation (LDA) and pass through three other approaches: labeled Embedded Topic Model (ETM), Gaussian LDA (G-LDA), and finally LDA with Word2Vec (LDA2Vec). These methods are the combination of LDA and word embeddings. We mention that there is another LDA method in supervised machine learning called Linear Discriminant Analysis. It is used for classification problems in statistics and other fields.

2.1 Latent Dirichlet Allocation (LDA)

LDA is a Bayesian generative probabilistic model that is used to discover the abstract topics that occur in a collection of documents. It was originally proposed by Blei et al. (2003).

The main idea behind LDA is that each document can be described by a distribution of topics and each topic can be described by a distribution of words (Figure 2.1).

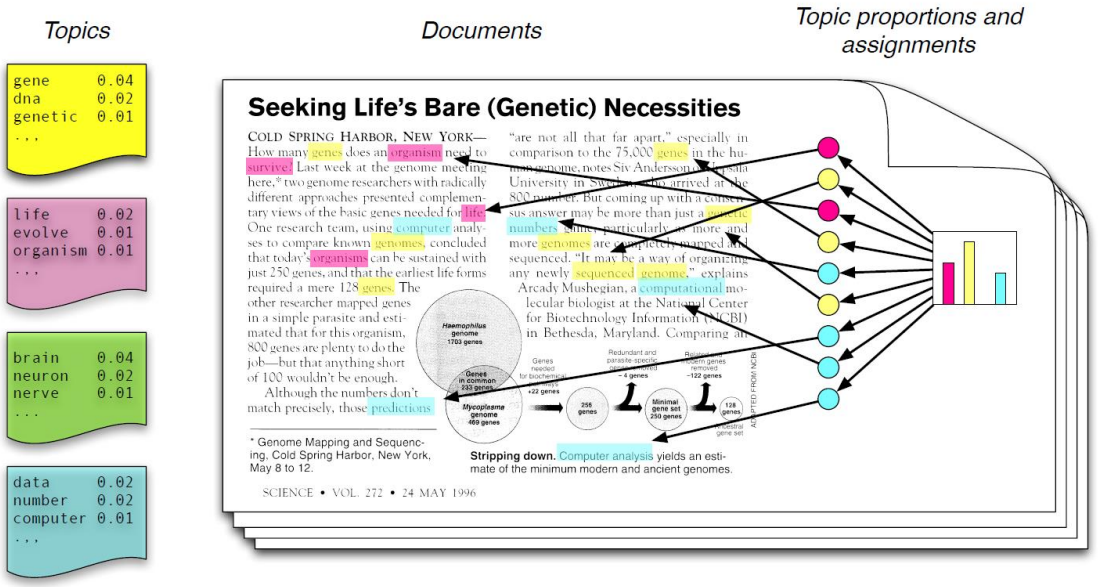


Figure 2.1: The intuition behind LDA: (Right) each document is a probability distribution over topics. (Left) each topic is a probability distribution over words (Blei et al. (2010)).

Let us say we have a collection of different newspapers (our corpus of documents), and we assume that we have a fixed number of topics within our corpus. Our goal is to find out what they are! To get there we make two key assumptions:

- Each document is a mixture of small number of topics.
- Each topic is a mixture of small number of words.

The intuition is that documents are similar if they have similar content. Further, when we see that some documents are connected to same set of words, we know they discuss the same topic. This idea is based on the distribution hypothesis, which says: words that appear together frequently are likely to be close in meaning. Therefore, what LDA does is connect the words to the topics depending on how well each word fall in a given topic, and then connect the topics to the documents based on what topics each document touch upon.

We note that LDA does not give importance to the order of words or the semantic in the document. Usually, LDA uses the bag-of-words feature representation to represent a document. That makes sense, because, when we take a document and jumble its words, we can still extract the topics that are discussed in the document.

We can describe the formal task of LDA as; given a collection of text documents as bag-of-words:
n_{wd} is a count of the word w in the document d.
The model trains to find:

- Probability distribution of words for each topic: $\phi_{wt} = p(w \mid t)$.
- Probability distribution of topics for each document: $\theta_{td} = p(t \mid d)$.

Before diving into the details, we present some definitions and notations.

Definitions and notations

- K: total number of topics.
- D: total number of documents in the corpus.
- N: total number of words in a given document.
- V: vocabulary size.
- α: hyper parameter for θ; encouraging sparseness of document-topics distribution (small number of topics associated with each document).
- β: hyper parameter for ϕ; encouraging sparseness of topic-words distribution (small number of words associated with each topic).
- ϕ_k: per-topic distribution over words, $K \times V$ matrix.

- θ_d: per-document distribution over topics, $D \times K$ matrix (one row for each document from the corpus).
- $z_{d,n}$: per-word topic assignment, $N \times K$ matrix (one row for each word from the document).
- $w_{d,n}$: observed word drawn from topic's word distribution (the n^{th} word in the d^{th} document).

To recap the aforementioned, α and β are Dirichlet-prior parameters, ϕ and θ are Dirichlet distributions, z and w are multinomials.

Figure 2.2 represents the graphical model of LDA.

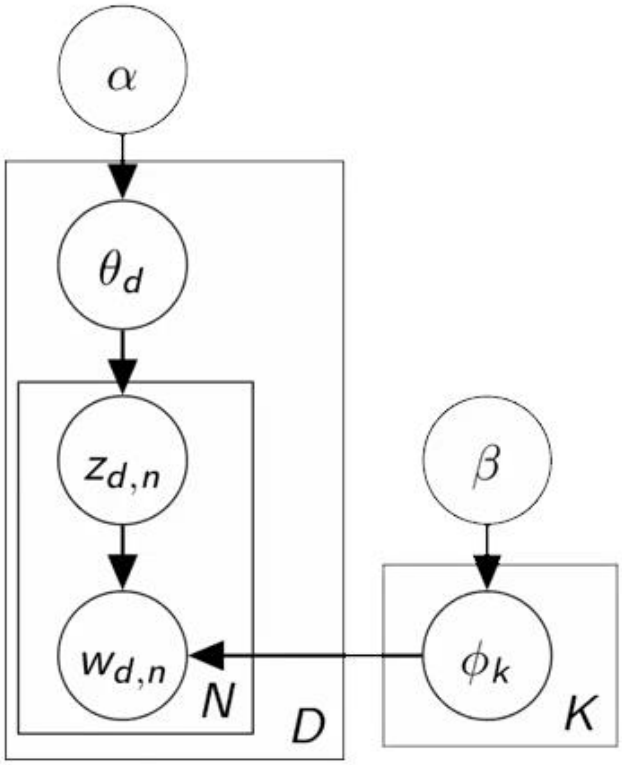


Figure 2.2: LDA graphical model (Blei and Lafferty (2009)).

Based on the previous definitions and notations, we present the corresponding mathematical formula for LDA:

$$p(\phi, \theta, z, w \mid \alpha, \beta) = \prod_{k=1}^{K} p(\phi_k \mid \beta) \prod_{d=1}^{D} p(\theta_d \mid \alpha) \prod_{n=1}^{N} p(z_{d,n} \mid \theta_d) p(w_{d,n} \mid \phi_{1:k}, z_{d,n}).$$

The generative process of LDA model (Algorithm 1):

Algorithm 1: Generative process of LDA model (Blei and Lafferty (2009))

for each topic $k \in \{1, \ldots, K\}$ **do**
    Choose a distribution over words: $\phi_k \sim Dir(\beta)$
for each document $d \in \{1, \ldots, D\}$ **do**
    Choose a distribution over topics: $\theta_d \sim Dir(\alpha)$
    for each word $n \in \{1, \ldots, N\}$ inside document d **do**
        Choose a topic from document's distribution: $z_{d,n} \sim Mult(\theta_d)$
        Choose a word from topic's distribution: $w_{d,n} \sim Mult(\phi_{z_{d,n}})$

2.2 Topic Modeling with LDA and Word Embeddings

LDA is a powerful model and it is used very widely in topic modeling. However, it suffers from some limitations, but we are going to focus on a particular problem and that is when we treat every word type (term) as a distinct entity. To clarify a little bit: each topic is a categorical distribution over the terms of our vocabulary. Therefore, the model learns a separate probability for each term, then we have a separate parameter for the probability of the word "cat" and the word "kitten" given a particular topic, i.e., there is no sharing between these distributions, although they are synonyms.

$$p(cat \mid t) \neq p(kitten \mid t).$$

The problem here is that if we have a corpus where we see the word "cat" a lot in our documents, but we hardly see the word "kitten". Consequently, the model learns very little about the word "kitten", i.e., it does not recognize which topic it is associated with. This problem appears when we have rare terms (terms that we do not see much in our training corpus) because we have unreliable signal for associating that word with a particular topic.

An alternative solution for this problem is word embeddings. The idea of word embeddings came in parallel with LDA model (Dieng et al. (2020)). The research started with a neural probabilistic language model that was proposed by Bengio et al. (2003), published in the same year and journal as LDA model, which was itself, proposed by Blei et al. (2003). Since then researchers have been extending and developing word embeddings, and nowadays it is considered as a standard technique in natural language processing.

Word embedding technique has become crucial in many applications of NLP sutch as sentiment analysis (Maas et al. (2011); Dos Santos and Gatti (2014)), text classification (Ahmad and Amin (2016); Zhao et al. (2018)), text extraction (Zeng et al. (2015)), machine translation (Bahdanau et al. (2014)), etc. This technique is capable to capture the context of a word in a document, relation with other words, etc. The idea is instead of modeling words we model continuous vector-space embeddings. These are feature representations for text where we can think of words as vectors in a vector space, and the words that have the same meaning have a similar representation. In other words, if the words are similar then they have similar features and they end up close together in the vector space as shown in Figure 2.3.

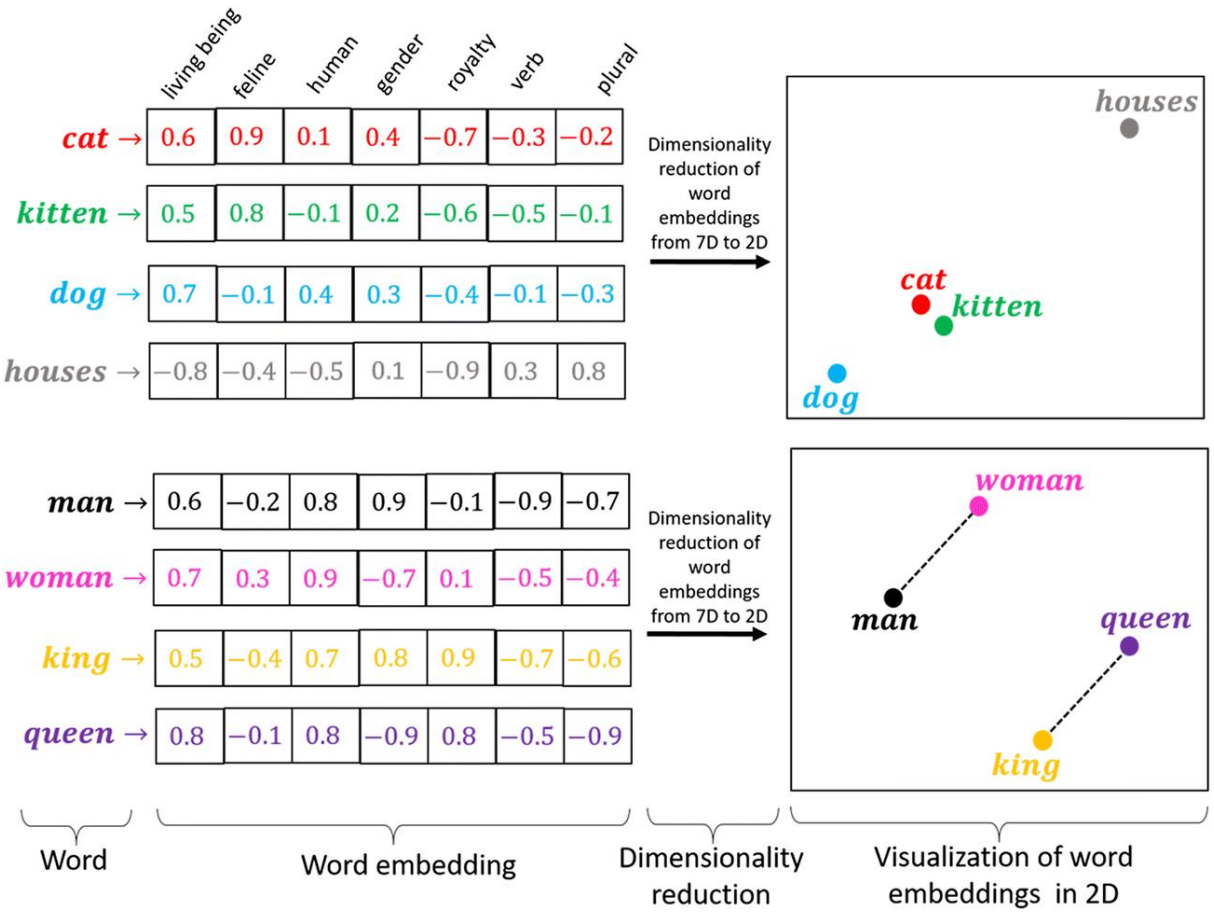


Figure 2.3: Word embedding examples.

The idea of training these embeddings is that, the features are based on the context in which a word gets used in the training corpus, and it exploits what is called the distributional hypothesis, which is essentially the words that tend to appear in similar contexts (in terms of words appearing around them) are likely to have a similar meaning.

Mixing LDA and word embeddings to gain powerful topic models

What we are going to do is instead of modeling the probability of words as we do in basic LDA, we are going to model the probability of the features associated with words that is the dimension of these embeddings. We can use pre-trained model of word embeddings that is trained on very large corpus, and that is easily available for English and some other languages.

The basic idea is that we train our embedding model on a large, general-domain corpus, and then we train our topic model on typically much smaller target corpus (the corpus that we are interested in modeling the topics of). We train this model in such a way that we take into account the information in the embeddings, and by doing this we benefit from the generalizations over word meanings that the embedding model make for us. For example, in case of the example we have mentioned earlier ("cat" and "kitten"), our model will learn from the embedding model that "cat" and "kitten" have a very similar meaning, and they get very similar features in the embedding space, and so it can then take advantage of that and learn similar probabilities for these words.

There are several topic models proposed by the researchers that make use of this idea. In this thesis, we are going to talk through a few of them. They are all assuming the availability of some set of embeddings $E(w)$ that is trained on some broad-domain corpus so that it has an embedding for each word w.

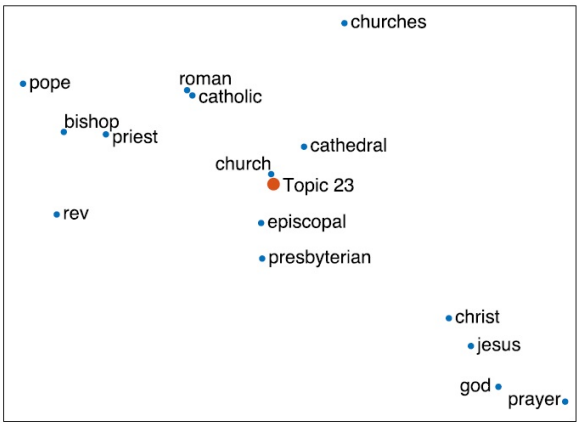
2.2.1 Embedded Topic Model (ETM)

ETM is a topic model that marries the probabilistic topic modeling of Latent Dirichlet Allocation (LDA) with the contextual information brought by word embeddings, most specifically, Word2Vec using CBOW model. It was originally published by Dieng et al. (2020). ETM benefits from the good properties of both topic models and word embeddings. As a topic model, it discovers the hidden topics inside documents; as a word embedding model, it provides the meaning of words. The result is a powerful document model that robustly accommodates large vocabularies including rare words, and maintains good performance over LDA.

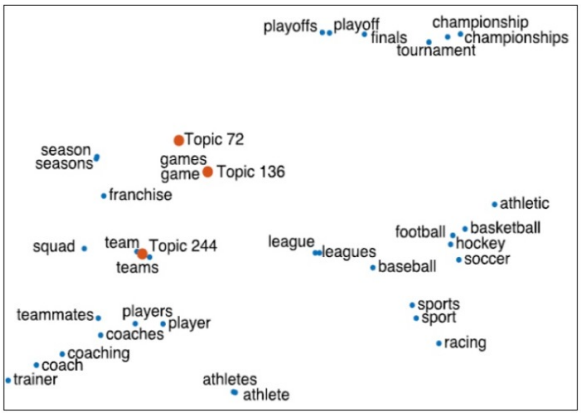
ETM is like LDA, a generative probabilistic model, where each document is a mixture of topics, and each observed word is assigned to a specific topic. However, unlike LDA, each word is defined by an embedding, and each topic is a point in the same embedding space Figure 2.4. Therefore, the per-topic distribution over words is relative to the exponentiated inner product of the topic's embedding and each word's embedding. In other words, the likelihood of a word

under ETM is a categorical whose natural parameter is given by the dot product between the word embedding and its assigned topic's embedding.

ETM models topics as points in the word embedding space, arranging together topics and words with similar context. Consequently, ETM can either learn word embeddings alongside topics or be given pre-trained embeddings to discover the topic patterns on the corpus.



(a) A topic about Christianity found by the ETM on $TheNewYorkTimes$. The topic is a point in the word embedding space.



(b) Topics about sports found by the ETM on The $NewYorkTimes$. Each topic is a point in the word embedding space.

Figure 2.4: ETM word embedding space. (Dieng et al. (2020))

We can describe the formal task of ETM as; given a corpus of D text documents containing V distinct words, and K latent topics.

The model trains to:

- Find the probability distribution of topics for each document θ_d.
- Embed each word in an L-dimensional space $E_v \in \mathbb{R}^L$.
- Embed each topic in an L-dimensional space $\alpha_k \in \mathbb{R}^L$.

ETM uses a log-linear model (softmax function) that takes as parameter the inner product of the word embedding matrix E and the topic's embedding α_k, to form a per-topic distribution over words, i.e., under ETM the assignment probability of a word to a specific topic is measured by the agreement between the word's embedding and the topic's embedding:

$$w_{d,n} \sim softmax(E^T \alpha_{d,n}).$$

Definitions and notations

- K: total number of topics.
- D: total number of documents in the corpus.
- N: total number of words in a given document.
- V: vocabulary size.
- L: dimensional space.
- E: embedding matrix of size $L \times V$ whose columns contain the embedding representations of the vocabulary; E_v is a vector of size L that contains the embedding of the word v.
- α_k: topic embedding vector of size L.
- θ_d: per-document distribution over topics, $D \times K$ matrix (one row for each document from the corpus).
- $z_{d,n}$: per-word topic assignment, $N \times K$ matrix (one row for each word from the document).
- $w_{d,n}$: observed word drawn from a distribution defined by the softmax of the dot product of the word embedding matrix E and the topic's embedding α_k (the n-th word in the d-th document).

ETM is similar in structure to LDA as shown in Figure 2.5.

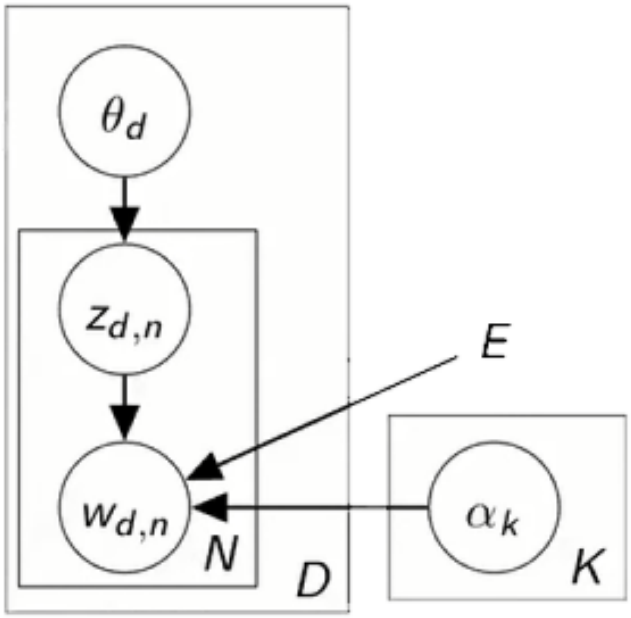


Figure 2.5: ETM graphical model.

The generative process of ETM model (Algorithm 2):

Algorithm 2: Generative process of ETM model (Dieng et al. (2020)).

1 **for** each document $d \in \{1, \ldots, D\}$ **do**
2 Choose a distribution over topics: $\theta_d \sim \mathcal{LN}(0, I)$
3 **for** each word $n \in \{1, \ldots, N\}$ inside document d **do**
4 Choose a topic from document's distribution: $z_{d,n} \sim Cat(\theta_d)$
5 Choose a word: $w_{d,n} \sim softmax(E^T \alpha_{z_{d,n}})$

We note that $\mathcal{LN}(.)$ in line 2, indicates the logistic-normal distribution (Blei and Lafferty (2007)). A drawn θ_d from this distribution is acquired as:

$$\delta_d \sim N(0, I); \qquad \theta_d \sim softmax(\delta_d).$$

2.2.2 Gaussian LDA (G-LDA)

LDA is a statistical method where the corpus is considered as the collection of a small number of topics and these topics are considered as a distribution of words. Although this method produces good results, however, as we know that documents are a mixture of topics; are these topics semantically coherent?

Topics in LDA are semantically incoherent and this is one of the Limitations of LDA model, according to (Newman et al. (2009); Chang et al. (2009)). In order to solve this problem, the G-LDA method that was originally proposed by Das et al. (2015) can find the semantic coherence between topics using word embeddings technique that has shown its efficacy in capturing the lexico-semantic regularities in language. In other terms, words with similar syntactic and semantic properties are close to each other in the embedding space (Mikolov et al. (2013); Agirre et al. (2009)).

G-LDA is a topic model and it is considered as an extension of LDA, which replaces categorical distributions over word in LDA with multivariate gaussian distributions over the word embedding space.

The main idea of G-LDA is that it replaces the opaque word modeled in LDA with continuous space embeddings of these words, which are generated as draws from a multivariate gaussian, as shown in Figure 2.6.

G-LDA considers topics K as multivariate gaussian distributions with mean μ_k and covariance Σ_k on the continuous embedding space distributions over words. It is based on the following values: a gaussian distribution μ_k centered at zero for the mean and an inverse wishart distribution for the covariance matrix Σ_k, where the inverse wishart is used as the conjugate prior for the covariance matrix of a multivariate normal distribution and it ensures positive definiteness of this matrix. The inverse wishart function was discovered for the first time by John Wishart in 1928, it defined with a matrix of $r \times r$ dimension and a number of degrees of freedom df, with $df > (r - 1)$ according to (Sawyer (2007)).

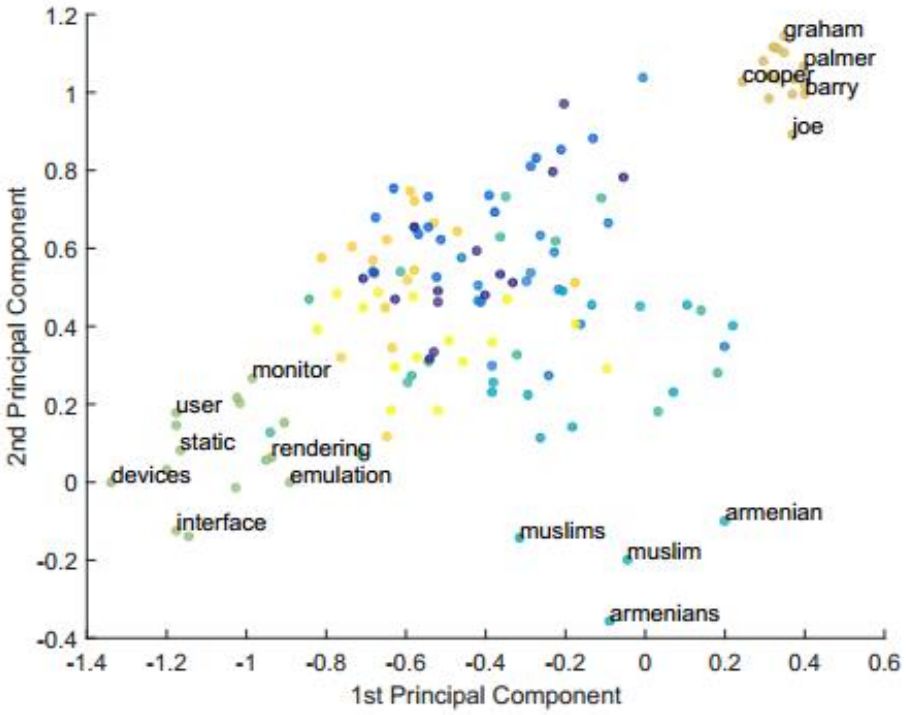


Figure 2.6: The first two principal components for the word embeddings some topics from Gaussian LDA on the 20 newsgroups dataset has been visualized. Each point represents a word color according to its topic (Das et al. (2015)).

Definitions and notations:

- K: total number of topics.
- D: total number of documents in the corpus.
- N: total number of words in a given document.
- θ_d: per-document distribution over topics.
- $v_{d,n}$: indexing a vector v of word (w) in a document d at position n.
- m, k, Ψ, ν: hyper parameters for gaussian topics .
- (μ_k, Σ_k): represents the mean and covariance of a multidimensional gaussian distribution.

Figure 2.7 represents the graphical model of G-LDA.

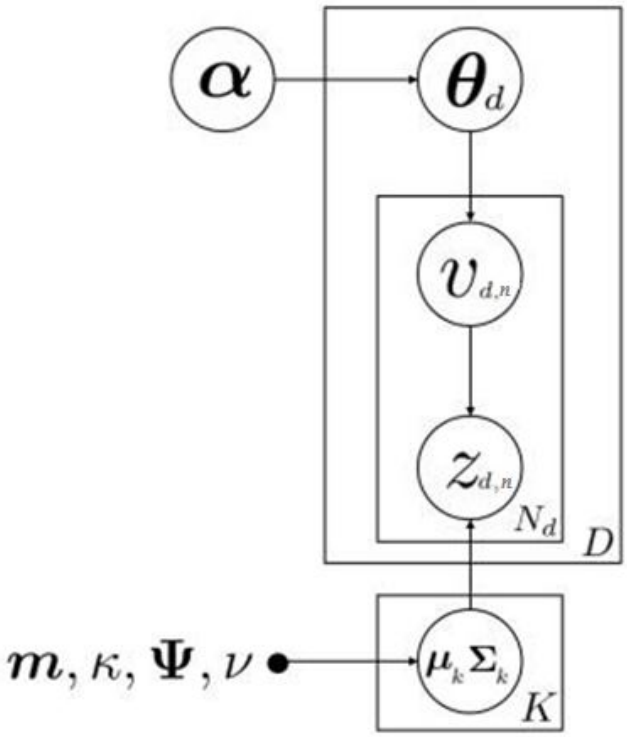


Figure 2.7: G-LDA graphical model (Ozaki and Kobayashi (2018)).

The generative process of G-LDA model (Algorithm 3):

Algorithm 3: Generative process of G-LDA model (Das et al. (2015))

for k = 1 to K **do**
  Draw topic covariance $\Sigma_k \sim W^{-1}(\Psi, \nu)$
  Draw topic mean $\mu_k \sim N(\mu, \frac{1}{k}\Sigma_k)$
for each document d in corpus D **do**
  Draw topic distribution $\theta_d \sim Dir(\alpha)$
  for each word index n from 1 to N_d **do**
    Draw a topic $z_n \sim Categorical(\theta_d)$
    Draw $v_{d,n} \sim N(\mu_{zn}, \Sigma_{zn})$

2.2.3 LDA with Word2Vec (LDA2Vec)

A topic model's main objective is to create interpretable document representations which are used to find topics in a large number of unlabeled documents. For example, in a document X we have 20% of topic A, and 40% of topic B and 40% of topic C.

LDA2Vec is a combination of LDA and Word2Vec model. It was originally published by Moody (2016). We know that the document is a mixture of topics, and the topic is a mixture of words. In the words level, Word2Vec is used to obtain vector representations of words.

LDA2Vec is based specifically on the Skip-Gram model of Word2Vec to generate word vectors, who trains the word embeddings using the input word to predict the words context. LDA2Vec utilizes a modified version of Skip-Gram named Skip-Gram Negative-Sampling (SGNS), objective to using document-wide feature vectors while simultaneously learning continuous document weights loading onto topic vectors.

LDA2Vec learns word vector in parallel with document vector (Figure 2.8). It makes the sum between them to predict the context vector (i.e., the context vector is generated from the sum of the two other vectors: the word and document vectors). Word vector is generated by Skip-Gram (see section 1.4.2), while the document vector is divided into the document weights vector and the topics matrix; the document weights vector represents the percentages of each topic in the document, calculated by the soft max function, which converts weights into percentages (see section 1.4.2), and the topics matrix represents the various topic vectors. Figure 2.8 represent LDA2Vec network architecture.

Skip-Gram Negative-Sampling (SGNS) uses the loss function to improve the quality of the results.

The loss function formula is as follows:

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

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

where:

$\mathcal{L}$: total loss.

(j, i): pairs of pivot and target words.

K: total number of topics.

$\vec{c_j}$: context vector.

$\vec{w_j}$: pivot word vector.

$\vec{w_i}$: target word vector.

$\vec{w_l}$: negatively-sampled word vector.

$\mathcal{L}^d$: Dirichlet-probability over document weights.

The $\mathcal{L}^d$ encourages document proportions vectors to become more concentrated (sparser) over time.
calculation formula of $\mathcal{L}^d$ is as follows:

$$\mathcal{L}^d = \lambda \sum_{jk} (\alpha - 1) log P_{jk}$$

where:

λ: tuning parameter.

K: total number of topics.

$\alpha = K^{-1}$.

P_{jk}: document weights.

(j, k): the document j in topic k.

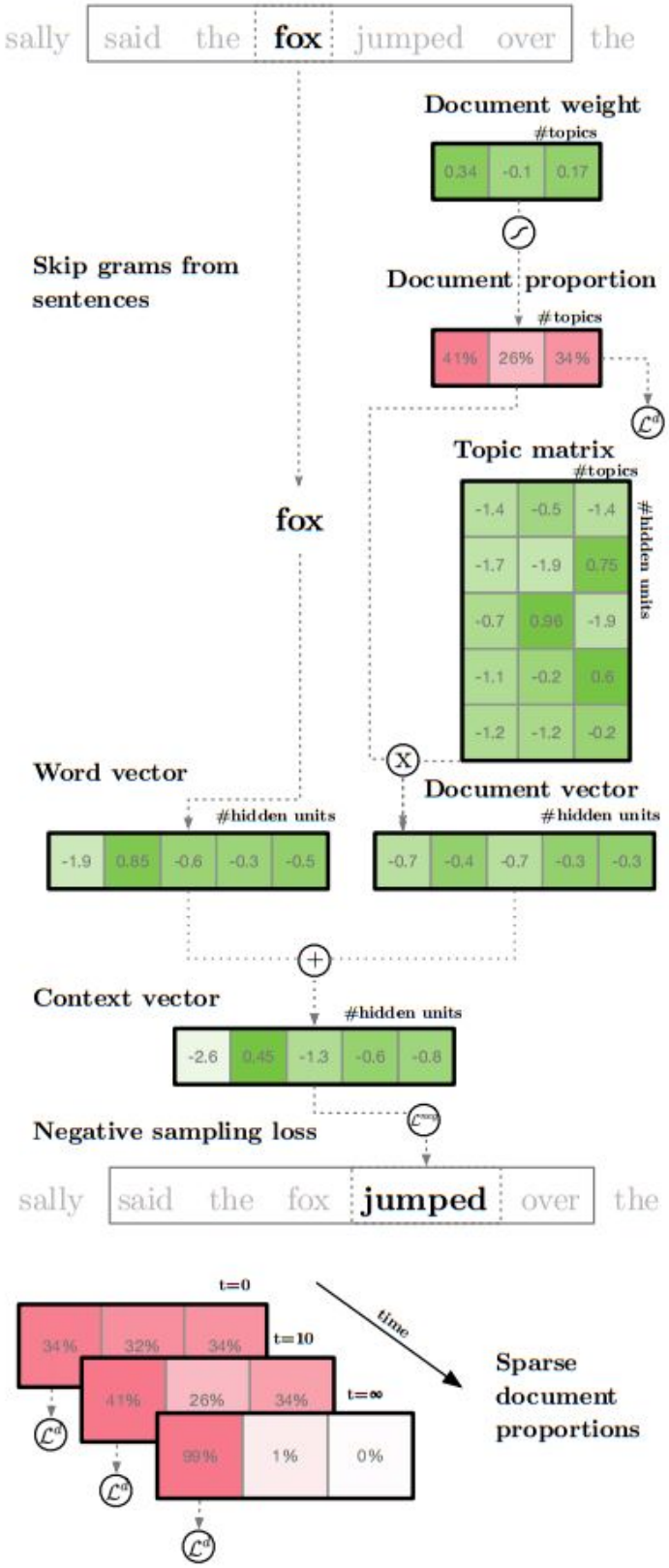


Figure 2.8: LDA2Vec network architecture (Moody (2016)).

Chapter 3

3 Implementation

3.1 Introduction

In this chapter, we carry out an experimental study that represents a comparison between two methods of topic modeling, namely LDA and ETM. The evaluation is carried out according to runtime and different topic coherence measures (Cv, UCI, NPMI, UMass).

We choose LDA because it is the most commonly used topic model, and it has a similar generative process as the ETM, we also considered ETM because it is a new document model technique that benefits from LDA properties and word embeddings, which makes it a powerful and an interesting model to work with in topic modeling. We apply this study on the 20Newsgroups dataset.

3.2 Environment

In this section, we present the software and hardware of the environment we used to develop our experimental study.

For implementation purpose, we choose Python as a programming language due to its efficient, robust and powerful performances in the field of natural language processing (NLP), most specifically, topic modeling. Moreover, Python provides several libraries that allow us to make our experiments in a very easy and efficient way.

Concerning needed packages, we use NLTK (Natural Language Toolkit) library for data preprocessing, more precisely, stop words removal, tokenization and lemmatization. As regards topic modeling, we use Gensim, an open-source library, specialized major in topic modeling, which has shown its efficiency and flexibility for constructing multiple text representations (e.g., Bag-of-Words and Word2Vec), and training multiple topic models, like LDA, on text

document corpora in order to extract semantic structures, and also evaluating topic models by calculating their topic coherence scores (e.g., Cv, UCI, UMass, NMPI). In connection with topic modeling, we use also embedded-topic-model package, that was made to easily run embedded topic modelling on a given corpus.

To train our models, we use Google Colab, a free cloud service provided by Google, that offers Jupyter Notebooks with easy sharing and allows users to write and execute Python. We use this platform for our experiments under the following features:

- 2.2 Ghz CPU.
- 12.72 GB of RAM.
- 107.77 GB of disk.

3.3 Data Preprocessing

The dataset we used in our experimental study is the 20 Newsgroups. This dataset is a collection of nearly about 18,000 newsgroup documents partitioned across 20 different newsgroups (Figure 3.1). It has become popular and widely used for experiments of machine learning techniques in text applications like topic modeling.

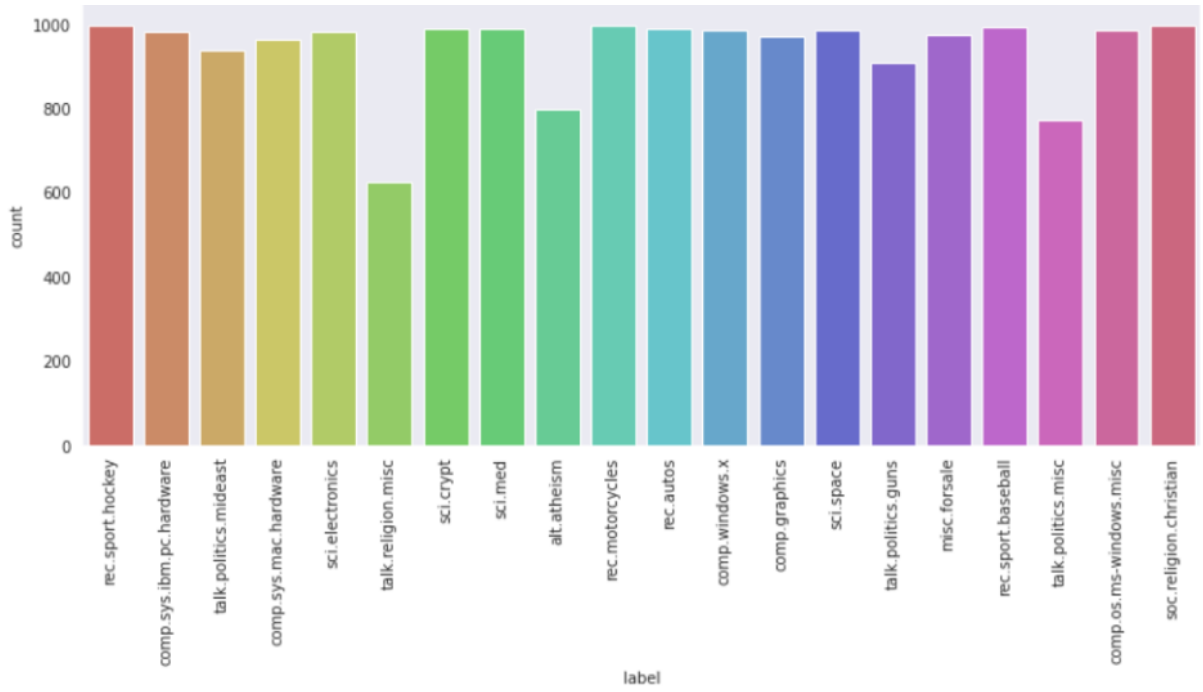


Figure 3.1: 20 newsgroups dataset.

We can upload the dataset from the scikit-learn library by importing the fetch_20newsgroups function from sklearn.datasets module (Listing 1), or we can download it manually from

Kaggle datasets (https://www.kaggle.com/crawford/20-newsgroups), and then upload it to the environment.

```
from sklearn.datasets import fetch_20newsgroups

full_dataset = fetch_20newsgroups(
    subset='all',
    remove=('headers', 'footers', 'quotes'),
    shuffle=True,
    random_state=42
)
data = pd.DataFrame()
data['text'] = full_dataset.data
data['source'] = full_dataset.target
label = []
for i in data['source']:
    label.append(full_dataset.target_names[i])
data['label'] = label
```

Listing 1: Importing dataset

Data preprocessing is a fundamental task in natural language processing applications like topic modeling. This process essentially is beneficial for obtaining good and more accurate results. To reach this goal, before diving into experiments, we accomplish the aforementioned with some preprocessing techniques on the corpus to prepare it for the topic models.

First of all, we clean the dataset from any non-alphabetic characters (numbers, punctuations, line breaks, etc.) using regular expressions (called REs, or regexes, or regex patterns). Therefore, Python has a built-in package called re, which is intended to handle regular expression operations, and provides full support for Perl programming language. We also proceed lower casing, deletion of words that have at most three characters, stop words removal and lemmatization (Listing 2).

```
import re, nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')

```

```
# cleaning dataset
def cleaned_text(text):
    clean = re.sub('\n', ' ', text)
    clean = clean.lower()
    clean = re.sub(r'[~.,%/:;?_&+*=!-]', ' ', clean)
    clean = re.sub('[^a-z]', ' ', clean)
    clean = clean.lstrip()
    return clean

data['cleaned_text'] = data['text'].apply(cleaned_text)

data['cleaned_text'] = data['cleaned_text'].apply(
    lambda x: ' '.join([word for word in x.split() if len(word)>3])
)

# stop words removal
stop = stopwords.words('english')
stop.append('also')
data['stopwords_removed_text'] = data['cleaned_text'].apply(
    lambda x: ' '.join([word for word in x.split() if word not in (stop)])
)

# tokenization
data['tokenized'] = data['stopwords_removed_text'].apply(
    lambda x: nltk.word_tokenize(x)
)

# lemmatization
def word_lemmatizer(text):
    lem_text = [WordNetLemmatizer().lemmatize(word, pos='v') for word in text]
    return lem_text

data['lemmatized'] = data['tokenized'].apply(
    lambda x: word_lemmatizer(x)
)

data['lemmatize_joined'] = data['lemmatized'].apply(
    lambda x: ' '.join(x)
)
```

Listing 2: Data preprocessing

Figure 3.2 shows the first and the last five documents of the corpus before and after data preprocessing using the mentioned operations.

doc 1	\n\nI am sure some bashers of Pens fans are pretty confused about the lack\nof any kind of posts...
doc 2	My brother is in the market for a high-performance video card that supports\nVESA local bus with...
doc 3	\n\n\n\n\tFinally you said what you dream about. Mediterranean???? That was new....\n\tThe area ...
doc 4	\nThink!\n\nIt's the SCSI card doing the DMA transfers NOT the disks...\n\nThe SCSI card can do ...
doc 5	1) I have an old Jasmine drive which I cannot use with my new system.\n My understanding is t...
...	...
doc 18842	DN> From: nyeda@cnsvax.uwec.edu (David Nye)\nDN> A neurology\nDN> consultation is cheaper than a...
doc 18843	\nNot in isolated ground recepticles (usually an unusual color, such as orange\nor yellow) often...
doc 18844	I just installed a DX2-66 CPU in a clone motherboard, and tried mounting a CPU \ncooler on the c...
doc 18845	\nWouldn't this require a hyper-sphere. In 3-space, 4 points over specifies\na sphere as far as...
doc 18846	After a tip from Gary Crum (crum@fcom.cc.utah.edu) I got on the Phone\nwith "Pontiac Systems" or...

(a) Corpus before preprocessing (original text).

doc 1	sure bashers pen fan pretty confuse lack kind post recent pen massacre devil actually puzzle rel...
doc 2	brother market high performance video card support vesa local anyone suggestions ideas diamond s...
doc 3	finally say dream mediterranean area greater years like holocaust number july sweden april still...
doc 4	think scsi card transfer disk scsi card transfer contain data scsi devices attach want important...
doc 5	jasmine drive can not system understand upsate driver modern order gain compatability system any...
...	...
doc 18842	nyeda cnsvax uwec david neurology consultation cheaper scan better neurologist make differential...
doc 18843	isolate grind recepticles usually unusual color orange yellow often use noise leakage applicatio...
doc 18844	instal clone motherboard try mount cooler chip hour weight cooler enough dislodge mount end bend...
doc 18845	require hyper sphere space point specify sphere unless prove point exist space equi distant poin...
doc 18846	gary crum crum fcom utah phone pontiac systems pontaic customer service whatever inquire rumour ...

(b) Corpus after preprocessing (text cleaning, lower casing, stop words removal and lemmatization).

Figure 3.2: The corpus before and after data preprocessing.

Secondly, we construct a dictionary that contains the vocabulary corresponding to all newsproup documents in the dataset, in order to encapsulate the mapping between each word from the vocabulary and its unique integer id. At this phase, the vocabulary size achieves 74,404 unique words (Listing 3).

```
1 from gensim import corpora
2
3 # Collection of docs
4 docs = data['lemmatized']
5
6 # Turn our tokenized documents into a dictionary
```

```
dictionary = corpora.Dictionary(docs)
```

Listing 3: Dictionary constructing

Thirdly, we filter out the words in the dictionary by their frequency to remove the words that they are not valuable in identifying the topics from documents. We apply this idea by eliminating all the words that rarely appear in the corpus (words that are contained in less than 50 documents). In the other hand, we also exclude all the words that appear very frequently in the corpus (words that are contained in more than 75% of documents). On the fulfillment of this task, the vocabulary size reduces to become 3,224 unique words (Listing 4).

```
# Filter out the rarely and the very frequently words
dictionary.filter_extremes(no_below=50, no_above=0.75)
```

Listing 4: Dictionary filtering

Finally, we create a document-term-matrix by converting each document from the corpus into a bag-of-words format (Listing 5).

```
# Convert tokenized documents into a document-term matrix
doc_term_matrix = [dictionary.doc2bow(doc) for doc in docs]
```

Listing 5: Document-term matrix constructing

3.4 Results and Discussion

After the implementation of both LDA and ETM models and data preprocessing, we finally get to the main task of our study, which is the comparison between the two mentioned topic models using the evaluation by multiple topic coherence measures (Cv, UCI, NPMI and UMass) and the runtime factor.

For each approach, we consider a different number of topics (100, 150, 200, 250, 300), and then we calculate the Cv, UCI, NPMI, UMass coherence scores, and finally the runtime.

3.4.1 LDA Evaluation

We start with training LDA model on the preprocessed corpus. To illustrate the process, we present an example considering the number of topics equals to 100 (Listing 6).

```
%%time

from gensim.models.ldamodel import LdaModel
```

```
4
5  # LDA model
6  lda_model = LdaModel(
7      corpus=doc_term_matrix,
8      id2word=dictionary,
9      num_topics=100,
10     passes=100,
11     chunksize=2500,
12     alpha='auto',
13     eta='auto'
14 )
```

Listing 6: LDA model

After training the model we calculate the desired topic coherence scores (Listing 7).

```
1  from gensim.models.coherencemodel import CoherenceModel
2
3  lda_topics = [
4      [term for term, wt in lda_model.show_topic(n, topn=20)]
5          for n in range(0, lda_model.num_topics)
6  ]
7
8  # Cv coherence
9  cv_coherence_model = CoherenceModel(
10     topics=lda_topics,
11     texts=docs,
12     dictionary=dictionary,
13     coherence='c_v'
14 )
15 cv_coherence = cv_coherence_model.get_coherence()
16
17 # UCI coherence
18 uci_coherence_model = CoherenceModel(
19     topics=lda_topics,
20     texts=docs,
21     dictionary=dictionary,
22     coherence='c_uci'
23 )
24 uci_coherence = uci_coherence_model.get_coherence()
25
```

```
# NPMI coherence
npmi_coherence_model = CoherenceModel(
    topics=lda_topics,
    texts=docs,
    dictionary=dictionary,
    coherence='c_npmi'
)
npmi_coherence = npmi_coherence_model.get_coherence()

# UMass coherence
umass_coherence_model = CoherenceModel(
    topics=lda_topics,
    corpus=doc_term_matrix,
    dictionary=dictionary,
    coherence='u_mass'
)
umass_coherence = umass_coherence_model.get_coherence()
```

Listing 7: LDA coherence scores

We repeat the same process considering each time new number of topics (100, 150, 200, 250, 300). The acquired results are displayed in Table 3.1.

Number of Topics	Cv Score	UCI Score	NPMI Score	UMass Score	Runtime Score
100	0.4892	-2.5879	-0.0416	-3.3551	1741
150	0.4447	-3.3807	-0.0943	-3.6061	1859
200	0.4173	-4.7760	-0.1333	-3.8150	2015
250	0.3982	-5.4577	-0.1637	-3.9794	2277
300	0.3900	-6.0864	-0.1898	-4.2436	2520

Table 3.1: LDA performance metrics.

3.4.2 ETM Evaluation

Before we run ETM model on the corpus, we need to pass through two important operations.

1- Pretrain the embeddings on the corpus to create a Word2Vec model or load a pretrained one (Listing 8). In this case, we must set the train_embeddings parameter to False when training the ETM instance (Listing 10).

In our experiments, we choose to work with Google's Word2Vec pre-trained model named Word2Vec-GoogleNews-vectors. The model was trained on a corpus of 3 billion words, and contains 3 million English word vectors of size 300-dimension (Listing 8 - Choice 2).

```
from embedded_topic_model.utils import embedding
import gensim.downloader as api

# Choice 1: Training word2vec embeddings on the corpus
documents = [doc for doc in data['cleaned_text']]
word_vectors = embedding.create_word2vec_embedding_from_dataset(documents)

# Choice 2: Load pre-trained word vectors from gensim-data
word_vectors = api.load('word2vec-google-news-300')
```

Listing 8: Word2Vec model

We can rather avoid the first operation, and instead learn the embeddings alongside topic modeling, by setting train_embeddings parameter to True (Listing 10).

2- Transform the corpus into a format understandable by the model (Listing 9).

```
from embedded_topic_model.utils import preprocessing

corpus = [doc for doc in data['lemmatize_joined']]

# Preprocessing the dataset
vocabulary, train_dataset, _, = preprocessing.create_etm_datasets(
    corpus,
    min_df=50,
    max_df=0.75,
    train_size=1
)
```

Listing 9: Corpus transformation for ETM instance

Now that we have the pretrained Word2Vec model, and the corpus transformed in a format understandable by the ETM instance, we can run the models on the preprocessed corpus. To illustrate the process, we present an example considering the number of topics equals to 100 (Listing 10).

```
%%time

from embedded_topic_model.models.etm import ETM

# Training an ETM instance
etm_model = ETM(
    vocabulary,
    embeddings=word_vectors,
    num_topics=100,
    epochs=100,
    debug_mode=True,
    train_embeddings=False,
)

# ETM model
etm_model.fit(train_dataset)
```

Listing 10: ETM model

After training the model we calculate the desired topic coherence scores (Listing 11).

```
from gensim.models.coherencemodel import CoherenceModel

etm_topics = etm_model.get_topics(20)

# Cv coherence
cv_coherence_model = CoherenceModel(
    topics=etm_topics,
    texts=docs,
    dictionary=dictionary,
    coherence='c_v'
)
cv_coherence = cv_coherence_model.get_coherence()

# UCI coherence
uci_coherence_model = CoherenceModel(
    topics=etm_topics,
```

```
    texts=docs,
    dictionary=dictionary,
    coherence='c_uci'
)
uci_coherence = uci_coherence_model.get_coherence()

# NPMI coherence
npmi_coherence_model = CoherenceModel(
    topics=etm_topics,
    texts=docs,
    dictionary=dictionary,
    coherence='c_npmi'
)
npmi_coherence = npmi_coherence_model.get_coherence()

# UMass coherence
umass_coherence_model = CoherenceModel(
    topics=etm_topics,
    corpus=doc_term_matrix,
    dictionary=dictionary,
    coherence='u_mass'
)
umass_coherence = umass_coherence_model.get_coherence()
```

Listing 11: ETM coherence scores

We repeat the same process considering each time new number of topics (100, 150, 200, 250, 300). The acquired results are displayed in Table 3.2.

Number of Topics	Cv Score	UCI Score	NPMI Score	UMass Score	Runtime Score
100	0.5049	-0.0376	0.0308	-2.1721	1366
150	0.4924	-0.1281	0.0250	-2.2050	1523
200	0.4854	-0.1522	0.0224	-2.2273	1686
250	0.4729	-0.3281	0.0147	-2.3120	1961
300	0.4799	-0.3630	0.0139	-2.3042	2204

Table 3.2: ETM performance metrics.

3.4.3 Comparison Between LDA and ETM

After the evaluation of LDA and ETM models, we compare the results to examine which approach dominates the other, i.e., we use the performance measures calculated earlier.

- Figure 3.3 displays the results concerning the Cv measure.
- Figure 3.4 displays the results concerning the UCI measure.
- Figure 3.5 displays the results concerning the NPMI measure.
- Figure 3.6 displays the results concerning the UMass measure.
- Figure 3.7 displays the results concerning the runtime measure.

In what follows, the five graphs that depict the behavior of LDA and ETM models concerning the performance measures over a specific number of topics. They give us a clear comparison between the approaches and a clear vision of which approach is best.

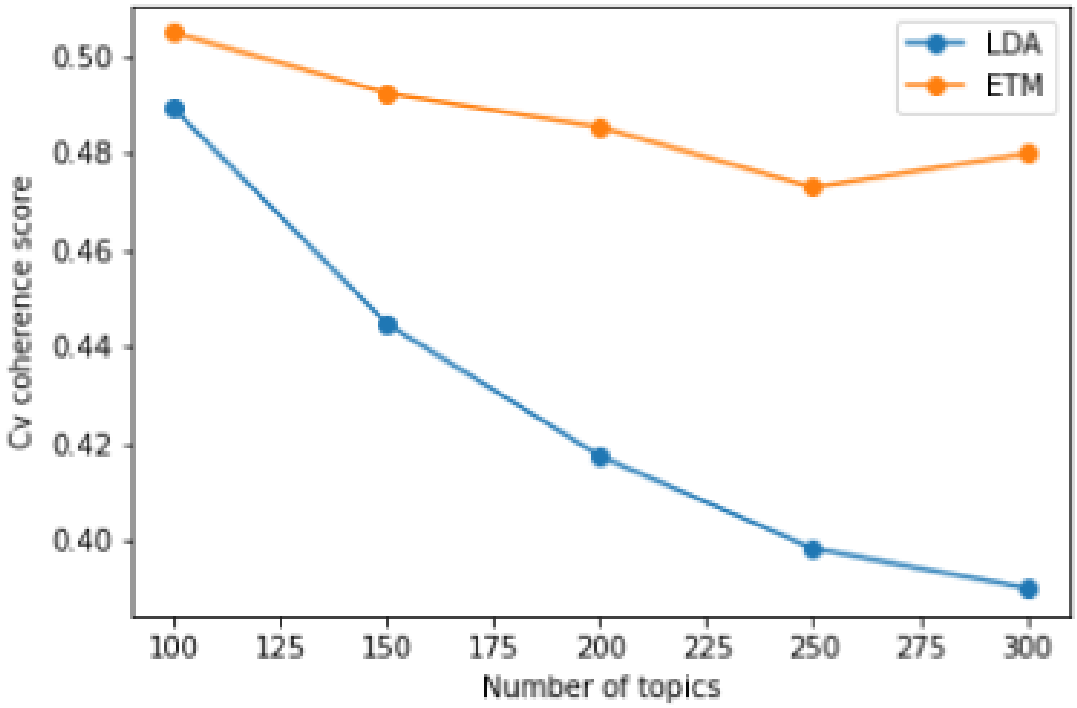


Figure 3.3: LDA vs ETM - Cv measure.

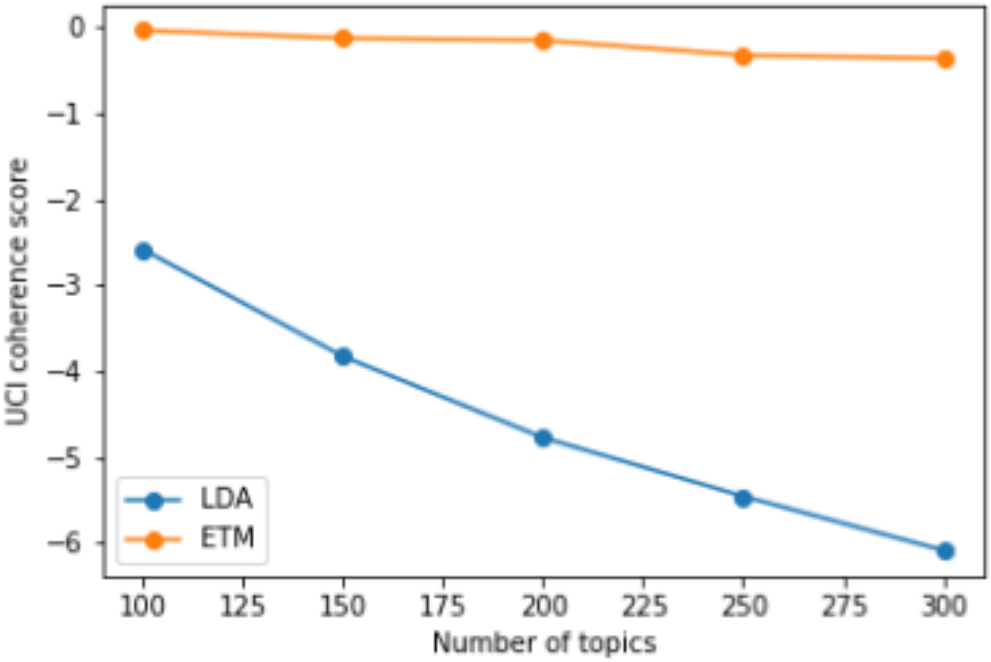


Figure 3.4: LDA vs ETM - UCI measure.

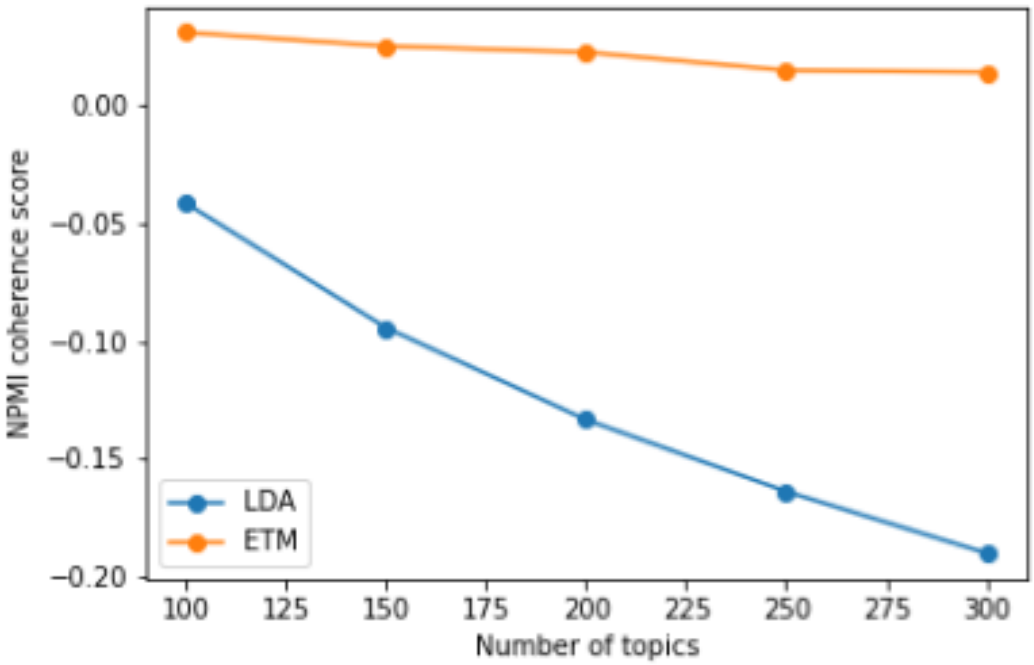


Figure 3.5: LDA vs ETM - NPMI measure.

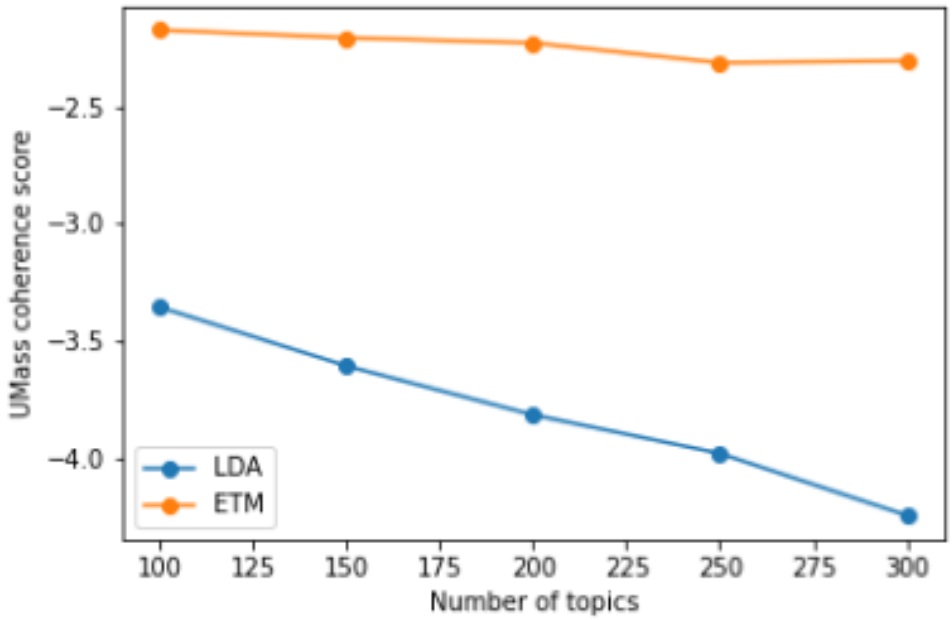


Figure 3.6: LDA vs ETM - UMass measure.

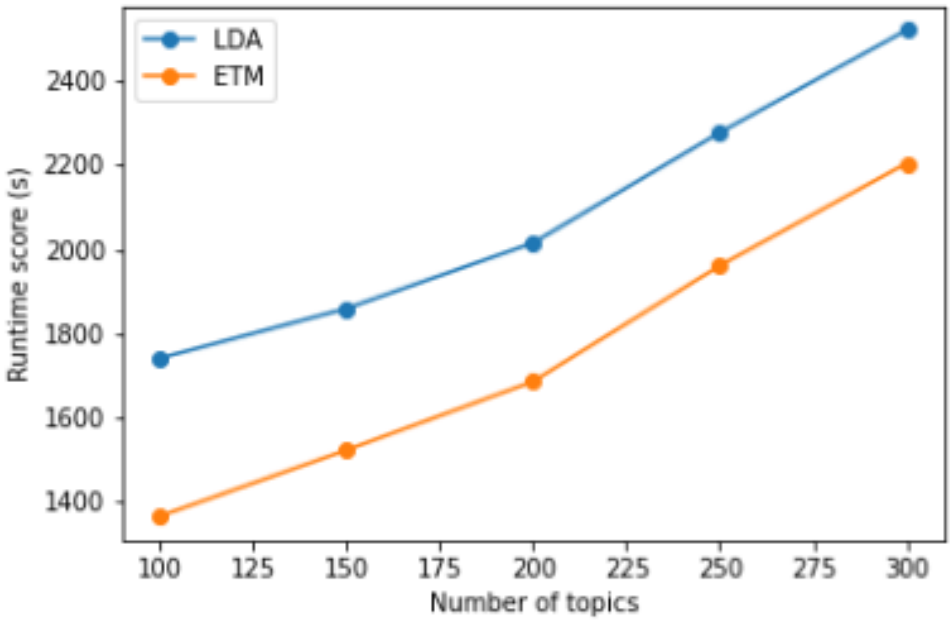


Figure 3.7: LDA vs ETM - runtime measure.

We can clearly observe that Cv (Figure 3.3), UCI (Figure 3.4), NPMI (Figure 3.5), and UMass scores (Figure 3.6) decrease with every increase in the number of topics for both LDA and ETM models.

Regarding the runtime (Figure 3.7), we can see that it is proportional to the number of topics, and it is obvious that ETM is better than LDA.

Considering all five measures we find that ETM absolutely outperforms LDA under the circumstances we performed our experimental study. We clarify the obtained result because of the semantic of words that are provided in the ETM model thanks to the word embedding technique (in our study, Word2Vec model). In opposite to the LDA model, which provides no support to the semantic of words.

4 Conclusion

Our thesis focused on an unsupervised machine learning technique called Topic Modeling, also classified as a Natural Language Processing (NLP) technique. This technique is meant for extracting the hidden topics discussed in a collection of documents (corpus).

In our study, we demonstrated a standard approach in the domain named Latent Dirichlet Allocation (LDA), and three other approaches referenced as Embedded Topic Model (ETM), Gaussian LDA (G-LDA), and LDA with Word2Vec (LDA2Vec). These approaches are the combination between LDA and the word embedding technique.

In this thesis, we performed a comparative study between LDA and ETM approaches. We used the 20 newsgroups dataset as a corpus, and Google's pre-trained Word2Vec as a word embedding model. The results we got in terms of runtime and topic coherence measures were all in favor of the ETM approach.

Word embedding is a powerful technique that helped with the tremendous improvement of Natural Language Processing (NLP) in general and topic modeling in particular.

As future suggestions, we would like to work on an Arabic corpus, and also perform the study in larger corpora in order to get more efficient and accurate results. We would like also to create a multiple languages topic model that is robust to noise.

References

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches.

Ahmad, A. and Amin, M. R. (2016). Bengali word embeddings and it's application in solving document classification problem. In *2016 19th International Conference on Computer and Information Technology (ICCIT)*, pages 425–430. IEEE.

Aletras, N. and Stevenson, M. (2013). Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 13–22.

Allen, C. and Hospedales, T. (2019). Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231. PMLR.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Blei, D., Carin, L., and Dunson, D. (2010). Probabilistic topic models. *IEEE signal processing magazine*, 27(6):55–65.

Blei, D. M. and Lafferty, J. D. (2007). A correlated topic model of science. *The annals of applied statistics*, 1(1):17–35.

Blei, D. M. and Lafferty, J. D. (2009). Topic models. In *Text mining*, pages 101–124. Chapman and Hall/CRC.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., and Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.

Das, R., Zaheer, M., and Dyer, C. (2015). Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804.

Dieng, A. B., Ruiz, F. J., and Blei, D. M. (2020). Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Dos Santos, C. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Ethayarajh, K., Duvenaud, D., and Hirst, G. (2019). Towards understanding linear word

analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Kim, H. K., Kim, H., and Cho, S. (2017). Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*, 266:336–352.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Moody, C. E. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv:1605.02019*.

Newman, D., Karimi, S., and Cavedon, L. (2009). External evaluation of topic models. In *in Australasian Doc. Comp. Symp., 2009*. Citeseer.

Newman, D., Lau, J. H., Grieser, K., and Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 100–108.

Ozaki, K. and Kobayashi, I. (2018). A study on stochastic variational inference for topic modeling with word embeddings.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Rani, S. and Kumar, M. (2021). Topic modeling and its applications in materials science and engineering. *Materials Today: Proceedings*, 45:5591–5596.

Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Sawyer, S. (2007). Wishart distributions and inverse wishart sampling. *URL: www. math. wustl. edu/sawyer/hmhandouts/Whishart. pdf*.

Vijayarani, S., Ilamathi, M. J., Nithya, M., et al. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16.

Zeng, D., Liu, K., Chen, Y., and Zhao, J. (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1753–1762.

Zhao, J., Zhou, Y., Li, Z., Wang, W., and Chang, K.-W. (2018). Learning gender-neutral word embeddings. *arXiv preprint arXiv:1809.01496*.