|  |  |  |
| --- | --- | --- |
|  | UNIVERSITY OF CRAIOVA  FACULTY OF AUTOMATION, COMPUTERS AND ELECTRONICS  COMPUTERS AND INFORMATION TECHNOLOGY DEPARTMENT |  |

DEGREE PROJECT

**Turcu Gabriel-Virgil**

|  |  |
| --- | --- |
| SCIENTIFIC COORDINATOR  **Assoc. Prof. Cristian Mihăescu, PhD.**  Department of Computer Science and Information Technoligies,  University of Craiova (Romania). | SCIENTIFIC COORDINATOR  **Dr. Javier Palanca**  Department of Computer Systems and Computation.  Universitat Politècnica de València (Spain). |

**July 2019**

**CRAIOVA**

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITY OF CRAIOVA  FACULTY OF AUTOMATION, COMPUTERS AND ELECTRONICS  COMPUTERS AND INFORMATION TECHNOLOGY DEPARTMENT |  |

**EduTranscriptMiner: LDA Based Information Retrieval System for**

**Educational Transcripts**

Turcu Gabriel-Virgil

|  |  |
| --- | --- |
| SCIENTIFIC COORDINATOR  **Assoc. Prof. Cristian Mihăescu, PhD.** | SCIENTIFIC COORDINATOR  **Dr. Javier Palanca** |

**July 2019**

**CRAIOVA**

*“Education is learning what you didn’t even know you didn’t know.”*

Daniel J. Boorstin

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul Turcu Gabriel-Virgil, student la specializarea Calculatoare Engleza din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul „EduTranscriptMiner: LDA Based Information Retrieval System for Educational Transcripts”
* coordonată de Conf. Univ. Dr. Ing. Cristian Mihăescu,
* prezentată în sesiunea Iulie 2019.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
* redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
* prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
* precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
* precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației | Aprobat la data de  …………………  Şef de departament,  Prof. dr. ing.  Marius BREZOVAN |

**PROIECTUL DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului: | Turcu Gabriel-Virgil |
| Enunțul temei: | EduTranscriptMiner: LDA Based Information Retrieval System for Educational Transcripts |
| Datele de pornire: | Un fisier cu date de tip metadata despre 45000 de videoclipuri educationale de pe site-ul <https://media.upv.es/> |
| Conținutul proiectului: | Primul capitol al lucrării expune scopul, motivația, enunțul și obiectivele temei. Următorul capitol descrie tehnologiile și uneltele folosite la realizarea acestui proiect. Al treilea capitol descrie articole conexe. Capitolul patru descrie arhitectura software în detaliu, de la formarea BoW și încorporarea cuvintelor folosind NNLM la prelucrarea unei interogări. Al cincilea capitol prezintă rezultatele experimentale după care în capitolul șase, lucrarea se încheie cu un capitol de concluzii și viitoarele îmbunătățiri. |
| Material grafic obligatoriu: | Schema Arhitecturala  Imagini cu interfața grafica |
| Consultații: | Periodice |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Conf. Univ. Dr. Ing. Cristian Mihăescu |
| Data eliberării temei: |  |
| Termenul estimat de predare a proiectului: |  |
| Data predării proiectului de către student și semnătura acestuia: |  |

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației |  |

**REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC**

|  |  |
| --- | --- |
| Numele și prenumele candidatului: | Turcu Gabriel-Virgil |
| Specializarea: | Calculatoare Engleza |
| Titlul proiectului: | EduTranscriptMiner: LDA Based Information Retrieval System for Educational Transcripts |
| Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): | În facultate □ |
| În producție □ |
| În cercetare □ |
| Altă locație: [*se detaliază*] |

În urma analizei lucrării candidatului au fost constatate următoarele:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nivelul documentării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Tipul proiectului | | Cercetare  □ | Proiectare  □ | Realizare practică □ | Altul  [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu  □ | Mediu  □ | Complex □ | Absent  □ |
| Utilitate | | Contract de cercetare □ | Cercetare internă □ | Utilare  □ | Altul  [*se detaliază*] |
| Redactarea lucrării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Partea grafică, desene | | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Realizarea practică | Contribuția autorului | Insuficientă  □ | Satisfăcătoare □ | Mare  □ | Foarte mare  □ |
| Complexitatea  temei | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Analiza cerințelor | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Arhitectura | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Întocmirea specificațiilor funcționale | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Implementarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Testarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Funcționarea | Da  □ | Parțială  □ | Nu  □ | |
| Rezultate experimentale | | Experiment propriu  □ | | Preluare din bibliografie  □ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii  și  observații | |  | | | |

În concluzie, se propune:

|  |  |
| --- | --- |
| ADMITEREA PROIECTULUI  □ | RESPINGEREA PROIECTULUI  □ |

Data, Semnătura conducătorului științific,

# Project Summary

Finding appropriate e-Learning resources within a repository of videos represents a critical aspect for students. Given that transcripts are available for the entire set of videos the problem reduces to obtaining a ranked list of video transcripts for a particular query. The paper presents a custom approach for searching the 16.012 available video transcripts from <https://media.upv.es/> at Universitat Politècnica de València.

An inherent difficulty of the problem comes from the fact that transcripts are in the Spanish language. The proposed solution embeds all the transcripts using feed-forward Neural-Net Language Models, clusters the embedded transcripts into 3 clusters to separate the transcripts into domains (Engineering & Architecture, Sciences (Biological Sciences), and Social & Legal) and builds a Latent Dirichlet Allocation (LDA) model for each cluster. We can then process a new query to find the cluster(domain) that it belongs to and then find the transcripts that have the LDA results closest to the LDA results for our query.

***Key Words****: Latent Dirichlet Allocation · NNLM word embeddings · Clustering*

***Mulțumiri***

Pe această cale mulțumesc persoanelor care m-au susținut și îndrumat pe parcursul elaborării acestei lucrări de liciență. Includ aici conducătorul științific care m-a îndrumat și sprijinit pentru a putea atinge scopul propus dar și familia care mi-a venit în ajutor pentru a mă susține moral.

Acest proiect a fost realizat într-o mobilitate Erasmus+ la Universitatea Politehnica din Valencia așa că doresc sa mulțumesc reprezentanților Universității Politehnice din Valencia pentru ca m-au acceptat în cadrul unei mobilități Erasmus de două luni la UPV și pentru îndrumarea oferită în cadrul proiectului. Mulțumesc pe aceasta cale și reprezentanților programului Erasmus+ pentru sprijinul financiar acordat prin program și pentru această oportunitate de a lucra într-un proiect internațional.

Mulțumesc în special domnului Conf. Dr. Ing Marian Cristian Mihăescu pentru îndrumarea dumnealui în cadrul acestui proiect dar nu numai.

Mulțumesc frumos pentru toată munca depusă și pentru tot suportul oferit în acești patru ani. Sfaturile și îndrumările dvs. m-au ajutat enorm în dezvoltarea profesionala cat și cea personala.

Fară îndrumările și ajutorul dumneavoastră nu aș fi ajuns la nivelul la care sunt acum. Iar pentru asta o să va fiu mereu recunoscător.

Contents

[Project Summary 9](#_Toc11888856)

[Tables List 13](#_Toc11888857)

[Figure List 14](#_Toc11888858)

[1 Introduction 15](#_Toc11888859)

[1.1 Purpose 15](#_Toc11888860)

[1.2 Motivation 16](#_Toc11888861)

[1.2.1 Machine Learning’s rise to popularity 16](#_Toc11888862)

[2 Domain Knowledge and Tools 18](#_Toc11888863)

[2.1 Python 18](#_Toc11888864)

[2.1.1 Features and philosophy 19](#_Toc11888865)

[2.1.2 Syntax and semantics 20](#_Toc11888866)

[2.1.3 Libraries 20](#_Toc11888867)

[2.1.4 Languages influenced by Python 21](#_Toc11888868)

[2.2 Machine Learning 23](#_Toc11888869)

[2.2.1 Types of problems and tasks 23](#_Toc11888870)

[2.2.2 Categorization of machine learning systems by outputs 24](#_Toc11888871)

[2.3 Scikit-learn 26](#_Toc11888872)

[2.4 K-means Clustering 27](#_Toc11888873)

[2.4.1 Description 27](#_Toc11888874)

[2.4.2 Initialization methods 28](#_Toc11888875)

[2.4.3 Complexity 29](#_Toc11888876)

[2.4.4 Variations 30](#_Toc11888877)

[2.4.5 Applications 31](#_Toc11888878)

[2.5 Gensim Library 33](#_Toc11888879)

[2.6 Latent Dirichlet allocation 34](#_Toc11888880)

[2.6.1 Topics 34](#_Toc11888881)

[2.6.2 Model 35](#_Toc11888882)

[2.6.3 Applications, extensions and similar techniques 37](#_Toc11888883)

[2.7 TensorFlow 39](#_Toc11888884)

[2.8 Neural-Net Language Models 40](#_Toc11888885)

[2.8.1 N-gram language models 42](#_Toc11888886)

[2.8.2 Distributed Representations 42](#_Toc11888887)

[3 Related Work 44](#_Toc11888888)

[4 System Design and Implementation 46](#_Toc11888889)

[4.1 Outline of Data Analysis Pipeline 46](#_Toc11888890)

[4.1.1 Build the bag-of-words 47](#_Toc11888891)

[4.1.2 Compute matrix with NNLM Word Embedding Results 48](#_Toc11888892)

[4.1.3 Build clusters of transcripts 49](#_Toc11888893)

[4.1.4 Model each cluster and find its topics and scores 50](#_Toc11888894)

[4.1.5 Query and retrieve a ranked list of results 52](#_Toc11888895)

[4.2 Graphical User Interface 54](#_Toc11888896)

[5 Experimental Results 56](#_Toc11888897)

[5.1 Input Dataset 56](#_Toc11888898)

[5.2 Numerical Results 58](#_Toc11888899)

[5.3 Validation 62](#_Toc11888900)

[6 Conclusions and Future Works 64](#_Toc11888901)

[7 References 65](#_Toc11888902)

# Tables List

[Table 1: Sample BoW with embedding results and cluster assignments. 59](#_Toc11888903)

[Table 2: Sample LDA models with their topics and scores 61](#_Toc11888904)

[Table 3: Query results. 61](#_Toc11888905)

[Table 4: TextRazor generated results for some queries 63](#_Toc11888906)

# Figure List

[Figure 1: Reinforcement Learning System 17](#_Toc11888907)

[Figure 2: Demonstration of the standard algorithm 29](#_Toc11888908)

[Figure 3: Vector quantization of colors present in the image on the left into Voronoi cells using k-means. 32](#_Toc11888909)

[Figure 4: Plate notation representing the LDA model. 35](#_Toc11888910)

[Figure 5: Plate notation for LDA with Dirichlet-distributed topic-word distributions 36](#_Toc11888911)

[Figure 6: Example of 2-dimensional distributed representation for words obtained 43](#_Toc11888912)

[Figure 7: Data Analysis Pipeline 46](#_Toc11888913)

[Figure 8: GUI Example 1 54](#_Toc11888914)

[Figure 9: GUI Example 2 54](#_Toc11888915)

# 1 Introduction

Searching for appropriate e-Learning resources (i.e., videos, quizzes, presentations, etc.) is one of the most critical activities for students that are willing to improve their knowledge. General purpose search engines may do an excellent job, but custom designed professional search tools are more advisable for better results.

Thus, within the area of e-Learning, the Video Base Learning (VBL) (Yousef, Chatti, & Schroeder, 2014) stands a particular place which gets more and more attention due to its effectiveness in teaching and learning. The significant technological advances have given the VBL a vital role in improving learning outcomes and properly designing VBL environments.

One critical aspect in VBL is the retrieval of relevant videos given an input query. This problem has been addressed in (Bakar, Kassim, Sahroni, & Anuar, 2017) by reviewing a wide range of machine learning algorithms that have been used for indexing and retrieving learning materials.

Among the most utilized indexing parameters, there are the ones that refer to text coming from natural language, documents or images. The various formats under which text may be shaped are web document, logs, XML, structured or semi-structured data. The general picture is completed by a wide range of indexing algorithms such as clustering, Ant colony, semantic index, SemTree, text index tree, B-Tree, etc.

The most critical shortcoming of search systems is that they are general and for existing implementations, they highly depend upon underlying data. Therefore, we have developed a specific search system over a dataset of video transcripts from a Spanish public university. A particularity of our input dataset consists of the fact that their transcripts are in the Spanish language. This poses new challenges as few developments that were done for the English language were also implemented for the Spanish language.

## Purpose

The objective of this project is to create a custom designed mechanism for indexing and retrieving video transcripts. The task is to index available video transcripts such that for an input query provided by a user the retrieval mechanism should return a list with the most representative videos. In our particular case, the input is represented by a large set of educational video transcripts and the search is accomplished by learners who are seeking learning materials.

The proposed approach takes as input 16.012 available video transcripts and builds a dataset by extracting necessary features. The dataset consists of a bag-of-words (BoW) and its corresponding matrix of NNLM embedding results. Given K, representing the number of domains which span the video transcripts, we run a clustering algorithm to obtain a partitioning. Thus, available video transcripts are assigned into clusters in an attempt to group by instances (i.e., video transcripts) into domains.

Once the domains are obtained, we further run an LDA (Blei, Ng, & Jordan, 2003) algorithm to get a list of topics and their score. Then, given a query, we determine the closest centroid and therefore obtain the domain of the query to which it belongs. Finally, by searching into the acquired domain’s instances, we get a ranked list of video transcripts that are closest to the query and return them to the user. Preliminary validation of the proposed solution has been performed manually by comparison with the outputs provided by the currently existing search mechanism.

## Motivation

My motivation behind deciding to work on this project was to learn a subject that has seen a massive growth in popularity over the last couple of years and that also seemed incredibly interesting to me, **Machine Learning**.

### Machine Learning’s rise to popularity

Machine Learning has started growing at an accelerating rate because nowadays we have cheap computational power as well as lots of data which we can use the train our models.

Recent years have shown that Machine Learning can be used to automate a lot of different tasks that where thought of as tasks that only humans can to like Image Recognition, Text Generation or playing games.

In 2014 Machine Learning and AI experts thought it would take at least 10 years before a machine could beat the world’s best player a t the board game Go. But Google’s DeepMind proved them wrong. They showed that even in such a complex game as Go machines could learn which moves to consider. There are a lot more of advances in the field of machines playing games like the Dota Bot from the OpenAI Team.

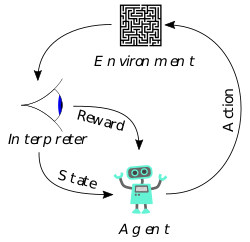


Figure 1: Reinforcement Learning System

Machine Learning is going to have huge effects on economy and living in general. Entire work tasks and industries can be automated and the job market will be changed forever.

Now that Machine Learning has all the attention it needs it’s on us the Engineers and Researchers to drive for new big advances in the field of Machine Learning. (TANNER, 2018)

# Domain Knowledge and Tools

For the completion of the degree project different domains, concepts, programming languages, libraries or platforms had to be studied, some of which I was first introduced to during this project.

## Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

### 2.1.1 Features and philosophy

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

### 2.1.2 Syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule.

Methods on objects are functions attached to the object's class; the syntax instance.method(argument)is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit *self* parameter to access instance data, in contrast to the implicit *self* (or *this*) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby)

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using **classes**, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass()or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.

### 2.1.3 Libraries

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333), but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over **130,000 packages** with a wide range of functionality, including:

* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Test frameworks
* Automation
* Web scraping
* Documentation
* System administration
* Scientific computing
* Text processing
* Image processing

### 2.1.4 Languages influenced by Python

Python's design and philosophy have influenced many other programming languages:

* ***Boo*** uses indentation, a similar syntax, and a similar object model.
* ***Cobra*** uses indentation and a similar syntax, and its "Acknowledgements" document lists Python first among languages that influenced it. However, Cobra directly supports design-by-contract, unit tests, and optional static typing.
* ***CoffeeScript***, a programming language that cross-compiles to JavaScript, has Python-inspired syntax.
* ***ECMAScript*** borrowed iterators and generators from Python.
* ***Go*** is designed for the "speed of working in a dynamic language like Python" and shares the same syntax for slicing arrays.
* ***Groovy*** was motivated by the desire to bring the Python design philosophy to Java.
* ***Julia*** was designed "with true macros [... and to be] as usable for general programming as Python [and] should be as fast as C". Calling to or from Julia is possible; to with *PyCall.jl* and a Python package *pyjulia* allows calling, in the other direction, from Python.
* ***Kotlin*** is a functional programming language with an interactive shell similar to Python. However, Kotlin is strongly typed with access to standard Java libraries.
* ***Ruby***'s creator, Yukihiro Matsumoto, has said: "I wanted a scripting language that was more powerful than ***Perl***, and more object-oriented than Python. That's why I decided to design my own language."
* ***Swift***, a programming language developed by Apple, has some Python-inspired syntax.
* ***GDScript***, dynamically typed programming language used to create video-games. It is extremely similar to Python with a few minor differences.

Python's development practices have also been emulated by other languages. For example, the practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python, a PEP) is also used in Tcl and Erlang.

Python received TIOBE's Programming Language of the Year awards in 2007, 2010 and 2018. The award is given to the language with the greatest growth in popularity over the year, as measured by the TIOBE index. (Python (programming language), n.d.)

## Machine Learning

“**Machine learning** is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machine learning as a "**Field of study that gives computers the ability to learn without being explicitly programmed**".

Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from an example training set of input observations in order to make data driven predictions or decisions expressed as outputs, rather than following strictly static program instructions.

Machine learning is closely related to (and often overlaps with) **computational statistics**; a discipline which also focuses in prediction making through the use of computers. It has strong ties to **mathematical optimization**, which delivers methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible.

Example applications include spam filtering, optical character recognition (OCR), search engines and computer vision. Machine learning is sometimes conflated with data mining, where the latter sub field focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

### 2.2.1 Types of problems and tasks

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:

* *Supervised learning*: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
* *Unsupervised learning*: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
* *Reinforcement learning*: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal. Another example is learning to play a game by playing against an opponent.

Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem instances is known at learning time, except that part of the targets are missing.

Among other categories of machine learning problems, “learning to learn” learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers, and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

### 2.2.2 Categorization of machine learning systems by outputs

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

* In **classification**, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi­label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
* In **regression**, also a supervised problem, the outputs are continuous rather than discrete.
* In **clustering**, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
* **Density estimation** finds the distribution of inputs in some space.
* **Dimensionality reduction** simplifies inputs by mapping them into a lower­-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

How well a model, trained with existing examples, predicts the output for unknown instances is called generalization. For best generalization, complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, we've underfitted. Then, we increase the complexity, the training error decreases. But if our hypothesis is too complex, we've overfitted. After then, we should find the hypothesis that has the minimum training error.

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

There are many similarities between machine learning theory and statistical inference, although they use different terms.” (Sebastian Thrun, Katie Malone, 2013)

## 2.3 Scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a *Google Summer of Code* project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from the French Institute for Research in Computer Science and Automation in Rocquencourt, France, took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012. As of 2018, scikit-learn is **under active development**.

Scikit-learn is largely written in **Python**, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. (Pedregosa, et al., 2011) (Cournapeau, 2019)

## 2.4 K-means Clustering

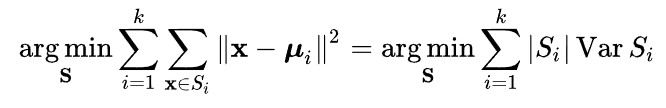
K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition **n** observations into **k** clusters in which each observation belongs to the cluster with the **nearest mean**, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

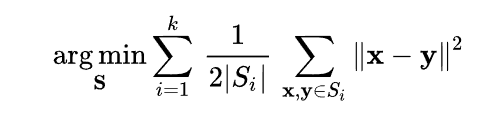
The algorithm has a loose relationship to the **k-nearest neighbor** classifier, a popular machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

### 2.4.1 Description

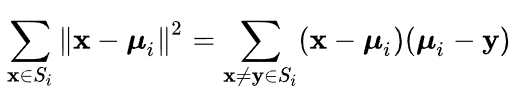
Given a set of observations (x**1**, x**2**, …, x**n**), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into **k** (≤ **n**) sets S = {S**1**, S**2**, …, S**k**} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:



where μi is the mean of points in Si. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:



The equivalence can be deduced from the following identity:



Because the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS), which follows from the law of total variance.

### 2.4.2 Initialization methods

Commonly used initialization methods are Forgy and Random Partition. The Forgy method randomly chooses k observations from the dataset and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points.

The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. The Random Partition method is generally preferable for algorithms such as the k-harmonic means and fuzzy k-means. For expectation maximization and standard k-means algorithms, the Forgy method of initialization is preferable. A comprehensive study, however, found that popular initialization methods such as Forgy, Random Partition, and Maximin often perform poorly, whereas Bradley and Fayyad's approach performs "consistently" in "the best group" and k-means++ performs "generally well".

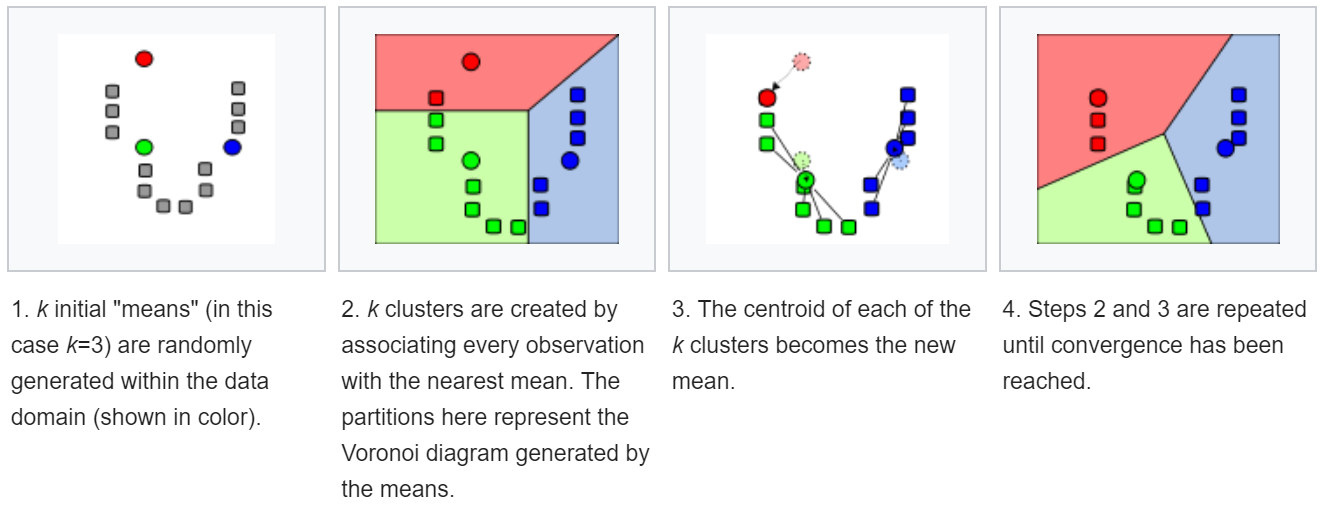


Figure 2: Demonstration of the standard algorithm

The algorithm does not guarantee convergence to the global optimum. The result may depend on the initial clusters. As the algorithm is usually fast, it is common to run it multiple times with different starting conditions. However, worst case performance can be slow: in particular certain point sets, even in 2 dimensions, converge in exponential time, that is 2Ω(n). These point sets do not seem to arise in practice: this is corroborated by the fact that the smoothed running time of k-means is polynomial.

The "assignment" step is referred to as the "expectation step", while the "update step" is a maximization step, making this algorithm a variant of the generalized expectation-maximization algorithm.

### 2.4.3 Complexity

Finding the optimal solution to the k-means clustering problem for observations in d dimensions is:

* NP-hard in general Euclidean space (of d dimensions) even for 2 clusters
* NP-hard for a general number of clusters k even in the plane
* if k and d (the dimension) are fixed, the problem can be exactly solved in time  where n is the number of entities to be clustered. Thus, a variety of heuristic algorithms such as Lloyd's algorithm given above are generally used.

The running time of Lloyd's algorithm (and most variants) is  where:

* n is the number of d-dimensional vectors (to be clustered)
* k the number of clusters
* i the number of iterations needed until convergence.

On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyd's algorithm is therefore often considered to be of "linear" complexity in practice, although it is in the worst-case super polynomial when performed until convergence.

In the worst-case, Lloyd's algorithm needs  iterations, so that the worst-case complexity of Lloyd's algorithm is super polynomial.

Lloyd's k-means algorithm has polynomial smoothed running time. It is shown that for arbitrary set of n points [0,1]d, if each point is independently perturbed by a normal distribution with mean 0 and variance б2, then the expected running time of k-means algorithm is bounded by , which is a polynomial in n, k, d and б.

Better bounds are proven for simple cases. For example, in it is shown that the running time of k-means algorithm is bounded by  for n points in an integer lattice {1…M}d.

Lloyd's algorithm is the standard approach for this problem. However, it spends a lot of processing time computing the distances between each of the k cluster centers and the n data points. Since points usually stay in the same clusters after a few iterations, much of this work is unnecessary, making the naive implementation very inefficient. Some implementations use caching and the triangle inequality in order to create bounds and accelerate Lloyd's algorithm.

### 2.4.4 Variations

* **Jenks natural breaks optimization**: k-means applied to univariate data
* **k-medians** clustering uses the median in each dimension instead of the mean, and this way minimizes L1 norm (Taxicab geometry).
* **k-medoids** (also: Partitioning Around Medoids, PAM) uses the medoid instead of the mean, and this way minimizes the sum of distances for arbitrary distance functions.
* **Fuzzy C-Means Clustering** is a soft version of k-means, where each data point has a fuzzy degree of belonging to each cluster.
* **Gaussian mixture** models trained with expectation-maximization algorithm (EM algorithm) maintains probabilistic assignments to clusters, instead of deterministic assignments, and multivariate Gaussian distributions instead of means.
* **K-means++** chooses initial centers in a way that gives a provable upper bound on the WCSS objective.
* **HG-means** is a sophisticated extension of K-means which escapes from local minima through iterative recombination steps and K-means improvement phases
* The filtering algorithm uses **kd-trees** to speed up each k-means step.
* Some methods attempt to speed up each k-means step using the triangle inequality.
* Escape local optima by swapping points between clusters.
* **The Spherical k-means** clustering algorithm is suitable for textual data.
* Hierarchical variants such as **Bisecting k-means, X-means** clustering and **G-means** clustering repeatedly split clusters to build a hierarchy, and can also try to automatically determine the optimal number of clusters in a dataset.
* Internal cluster evaluation measures such as cluster silhouette can be helpful at determining the number of clusters.
* **Minkowski weighted k-means** automatically calculates cluster specific feature weights, supporting the intuitive idea that a feature may have different degrees of relevance at different features. These weights can also be used to re-scale a given data set, increasing the likelihood of a cluster validity index to be optimized at the expected number of clusters.
* **Mini-batch k-means**: k-means variation using "mini batch" samples for data sets that do not fit into memory.

### 2.4.5 Applications

K-means clustering is rather easy to apply to even large data sets, particularly when using heuristics such as Lloyd's algorithm. It has been successfully used in market segmentation, computer vision, and astronomy among many other domains. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

* **Vector quantization**: K-means originates from signal processing, and still finds use in this domain. For example, in computer graphics, color quantization is the task of reducing the color palette of an image to a fixed number of colors k. The k-means algorithm can easily be used for this task and produces competitive results. A use case for this approach is image segmentation. Other uses of vector quantization include non-random sampling, as k-means can easily be used to choose k different but prototypical objects from a large data set for further analysis.

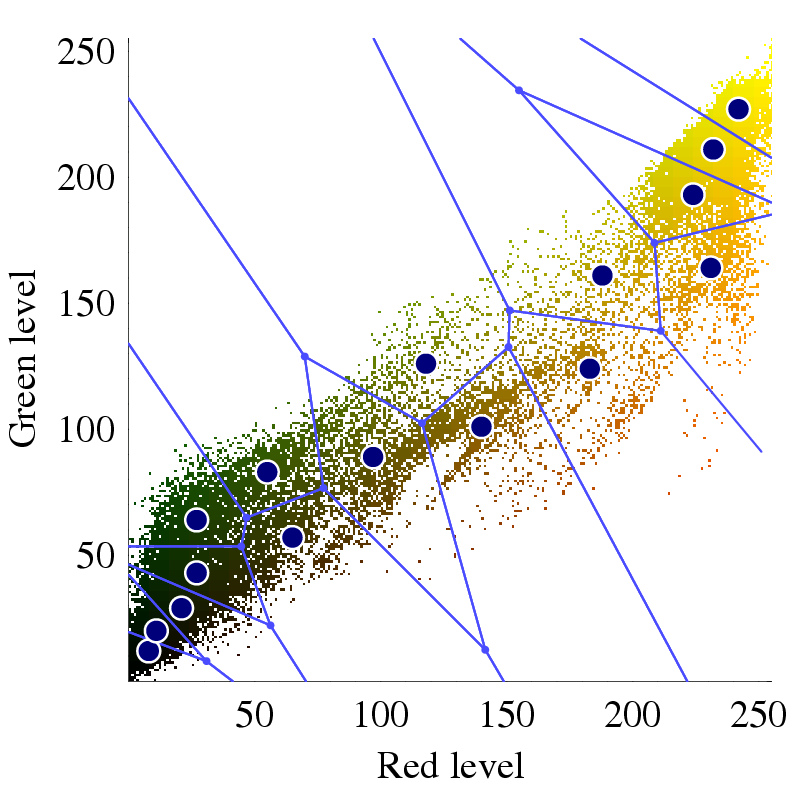


Figure 3: Vector quantization of colors present in the image on the left into Voronoi cells using k-means.

* **Cluster analysis**: In cluster analysis, the k-means algorithm can be used to partition the input data set into k partitions (clusters). However, the pure k-means algorithm is not very flexible, and as such is of limited use (except for when vector quantization as above is actually the desired use case). In particular, the parameter **k** is known to *be hard to choose* (as discussed above) when not given by external constraints. Another limitation is that it cannot be used with arbitrary distance functions or on non-numerical data. For these use cases, many other algorithms are superior.
* **Feature learning**: K-means clustering has been used as a feature learning (or dictionary learning) step, in either (semi-)supervised learning or unsupervised learning. The basic approach is first to train a k-means clustering representation, using the input training data (which need not be labelled). Then, to project any input datum into the new feature space, an "encoding" function, such as the thresholder matrix-product of the datum with the centroid locations, computes the distance from the datum to each centroid, or simply an indicator function for the nearest centroid, or some smooth transformation of the distance. Alternatively, transforming the sample-cluster distance through a Gaussian RBF, obtains the hidden layer of a radial basis function network. This use of k-means has been successfully combined with simple, linear classifiers for semi-supervised learning in NLP (specifically for named entity recognition) and in computer vision. On an object recognition task, it was found to exhibit comparable performance with more sophisticated feature learning approaches such as autoencoders and restricted Boltzmann machines. However, it generally requires more data, for equivalent performance, because each data point only contributes to one "feature". (K-means clustering, 2019)

## 2.5 Gensim Library

Gensim is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning.

Gensim is implemented in Python and Cython. Gensim is designed to handle large text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing.

Gensim started off as a collection of various Python scripts for the Czech Digital Mathematics Library **dml.cz** in 2008, where it served to generate a short list of the most similar articles to a given article (**gensim = “generate similar”**).

Gensim includes streamed parallelized implementations of ***fastText, word2vec and doc2vec algorithms***, as well as latent semantic analysis (LSA, LSI, SVD), non-negative matrix factorization (NMF), latent Dirichlet allocation (LDA), tf-idf and random projections.

Some of the novel online algorithms in Gensim were also published in the 2011 PhD dissertation Scalability of Semantic Analysis in Natural Language Processing of Radim Řehůřek, the creator of Gensim.

Gensim has been used and cited in over 1400 commercial and academic applications as of 2018, in a diverse array of disciplines from medicine to insurance claim analysis to patent search. The software has been covered in several new articles, podcasts and interviews.

The open source code is developed and hosted on GitHub and a public support forum is maintained on Google Groups and Gitter.

Gensim is commercially supported by the company rare-technologies.com, who also provide student mentorships and academic thesis projects for Gensim via their Student Incubator programme. (Řehůřek, 2019)

## 2.6 Latent Dirichlet allocation

In natural language processing, latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model.

### 2.6.1 Topics

In LDA, each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. LDA is a generalization of the pLSA model, which is equivalent to LDA under a uniform Dirichlet prior distribution.

For example, an LDA model might have topics that can be classified as CAT\_related and DOG\_related. A topic has probabilities of generating various words, such as milk, meow, and kitten, which can be classified and interpreted by the viewer as "CAT\_related". Naturally, the word cat itself will have high probability given this topic. The DOG\_related topic likewise has probabilities of generating each word: puppy, bark, and bone might have high probability. Words without special relevance, such as "the" (see function word), will have roughly even probability between classes (or can be placed into a separate category). A topic is neither semantically nor epistemologically strongly defined. It is identified on the basis of automatic detection of the likelihood of term co-occurrence. A lexical word may occur in several topics with a different probability, however, with a different typical set of neighboring words in each topic.

Each document is assumed to be characterized by a particular set of topics. This is similar to the standard bag of words model assumption, and makes the individual words exchangeable.

### 2.6.2 Model

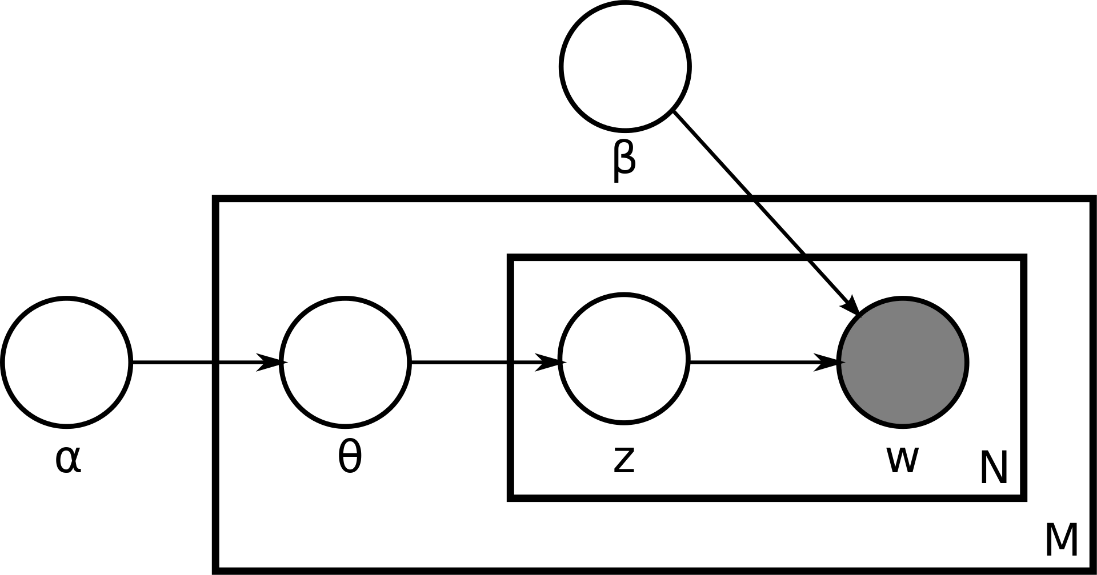


Figure 4: Plate notation representing the LDA model.



Figure 5: Plate notation for LDA with Dirichlet-distributed topic-word distributions

With plate notation, which is often used to represent probabilistic graphical models (PGMs), the dependencies among the many variables can be captured concisely. The boxes are "plates" representing replicates, which are repeated entities. The outer plate represents documents, while the inner plate represents the repeated word positions in a given document, each of which position is associated with a choice of topic and word. M denotes the number of documents, N the number of words in a document. The variable names are defined as follows:

* α is the parameter of the Dirichlet prior on the per-document topic distributions
* β is the parameter of the Dirichlet prior on the per-topic word distribution
* θi is the topic distribution for document **i**
* φk is the word distribution for topic **k**
* *z*ij is the topic for the **j**-th word in document **i**
* *w*ij is the specific word.

The fact that **W** is grayed out means that words ***w*ij** are the only observable variables, and the other variables are latent variables. As proposed in the original paper, a sparse Dirichlet prior can be used to model the topic-word distribution, following the intuition that the probability distribution over words in a topic is skewed, so that only a small set of words have high probability. The resulting model is the most widely applied variant of LDA today. The plate notation for this model is shown on the right, where **K** denotes the number of topics and **φ1… φK** are **V**-dimensional vectors storing the parameters of the Dirichlet-distributed topic-word distributions (**V** is the number of words in the vocabulary).

It is helpful to think of the entities represented by **θ** and **φ** as matrices created by decomposing the original document-word matrix that represents the corpus of documents being modeled. In this view, **θ** consists of rows defined by documents and columns defined by topics, while **φ** consists of rows defined by topics and columns defined by words. Thus, **φ1… φK** refers to a set of rows, or vectors, each of which is a distribution over words, and **θ 1… θ M** refers to a set of rows, each of which is a distribution over topics.

### 2.6.3 Applications, extensions and similar techniques

Topic modeling is a classic problem in information retrieval. Related models and techniques are, among others, latent semantic indexing, independent component analysis, probabilistic latent semantic indexing, non-negative matrix factorization, and Gamma-Poisson distribution.

The LDA model is highly modular and can therefore be easily extended. The main field of interest is modeling relations between topics. This is achieved by using another distribution on the simplex instead of the Dirichlet. The Correlated Topic Model follows this approach, inducing a correlation structure between topics by using the logistic normal distribution instead of the Dirichlet. Another extension is the hierarchical LDA (hLDA), where topics are joined together in a hierarchy by using the nested Chinese restaurant process, whose structure is learnt from data. LDA can also be extended to a corpus in which a document includes two types of information (e.g., words and names), as in the LDA-dual model. Nonparametric extensions of LDA include the hierarchical Dirichlet process mixture model, which allows the number of topics to be unbounded and learnt from data.

As noted earlier, pLSA is similar to LDA. The LDA model is essentially the Bayesian version of pLSA model. The Bayesian formulation tends to perform better on small datasets because Bayesian methods can avoid overfitting the data. For very large datasets, the results of the two models tend to converge. One difference is that pLSA uses a variable ***d*** to represent a document in the training set. So in pLSA, when presented with a document the model hasn't seen before, we fix ***Pr(w | z)*** the probability of words under topics to be that learned from the training set and use the same EM algorithm to infer ***Pr(z | d)*** - the topic distribution under **d**. Blei argues that this step is cheating because you are essentially refitting the model to the new data.

Variations on LDA have been used to automatically put natural images into categories, such as "bedroom" or "forest", by treating an image as a document, and small patches of the image as words; one of the variations is called Spatial Latent Dirichlet Allocation. (Blei, Ng, & Jordan, 2003) (Latent Dirichlet allocation, n.d.)

## 2.7 TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015. TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional **CUDA and SYCL** extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In Jan 2018, Google announced TensorFlow 2.0. In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript and TensorFlow Graphics for deep learning in computer graphics.

In May 2016, Google announced its Tensor Processing Unit (TPU), an application-specific integrated circuit (a hardware chip) built specifically for machine learning and tailored for TensorFlow. TPU is a programmable AI accelerator designed to provide high throughput of low-precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them. Google announced they had been running TPUs inside their data centers for more than a year, and had found them to deliver an order of magnitude better-optimized performance per watt for machine learning.

In May 2017, Google announced the second-generation, as well as the availability of the TPUs in Google Compute Engine (Yang & Meinel, 2014). The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs, provide up to 11.5 petaflops.

In May 2018, Google announced the third-generation TPUs delivering up to 420 teraflops of performance and 128 GB HBM. Cloud TPU v3 Pods offer 100+ petaflops of performance and 32 TB HBM. In February 2018, Google announced that they were making TPUs available in beta on the Google Cloud Platform.

Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as DeepDream. RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm-based search results. (Google Brain Team, n.d.)

## 2.8 Neural-Net Language Models

A **language model** is a function, or an algorithm for learning such a function, that captures the salient statistical characteristics of the distribution of sequences of words in a natural language, typically allowing one to make probabilistic predictions of the next word given preceding ones.

A **neural network language model** is a language model based on Neural Networks, exploiting their ability to learn distributed representations to reduce the impact of the curse of dimensionality.

In the context of learning algorithms, the curse of dimensionality refers to the need for huge numbers of training examples when learning highly complex functions. When the number of input variables increases, the number of required examples can grow exponentially. The curse of dimensionality arises when a huge number of different combinations of values of the input variables must be discriminated from each other, and the learning algorithm needs at least one example per relevant combination of values. In the context of language models, the problem comes from the huge number of possible sequences of words, e.g., with a sequence of 10 words taken from a vocabulary of **100,000** there are **1050** possible sequences...

A **distributed representation** of a symbol is a tuple (or vector) of features which characterize the meaning of the symbol, and are not mutually exclusive. If a human were to choose the features of a word, he might pick grammatical features like gender or plurality, as well as semantic features like animate or invisible. With a neural network language model, one relies on the learning algorithm to discover these features, and the features are continuous-valued (making the optimization problem involved in learning much simpler).

The basic idea is to learn to associate each word in the dictionary with a continuous-valued vector representation. Each word corresponds to a point in a feature space. One can imagine that each dimension of that space corresponds to a semantic or grammatical characteristic of words. The hope is that functionally similar words get to be closer to each other in that space, at least along some directions. A sequence of words can thus be transformed into a sequence of these learned feature vectors. The neural network learns to map that sequence of feature vectors to a prediction of interest, such as the probability distribution over the next word in the sequence. What pushes the learned word features to correspond to a form of semantic and grammatical similarity is that when two words are functionally similar, they can be replaced by one another in the same context, helping the neural network to compactly represent a function that makes good predictions on the training set, the set of word sequences used to train the model.

The advantage of this distributed representation approach is that it allows the model to generalize well to sequences that are not in the set of training word sequences, but that are similar in terms of their features, i.e., their distributed representation. Because neural networks tend to map nearby inputs to nearby outputs, the predictions corresponding to word sequences with similar features are mapped to similar predictions. Because many different combinations of feature values are possible, a very large set of possible meanings can be represented compactly, allowing a model with a comparatively small number of parameters to fit a large training set.

### 2.8.1 N-gram language models

The dominant methodology for probabilistic language modeling since the 1980's has been based on *n-gram* models. These non-parametric learning algorithms are based on storing and combining frequency counts of word subsequences of different lengths, e.g., 1, 2 and 3 for 3-grams. If a sequence of words ending in  ⋯*wt*−2,*wt*−1,*wt*,*wt*+1 is observed and has been seen frequently in the training set, one can estimate the probability in the training set   
*P*(*wt*+1|*w*1,⋯,*wt*−2,*wt*−1,*wt*) of *wt*+1 following *w*1,⋯*wt*−2,*wt*−1,*wt* by ignoring context beyond *n*−1 words, e.g., 2 words, and dividing the number of occurrences of *wt*−1,*wt*, *wt*+1 by the number of occurrences of *wt*−1,*wt*. Note that in doing so we ignore the identity of words that preceded *wt*−1. Furthermore, a new observed sequence typically will have occurred rarely or not at all in the training set. An important idea in n-grams is therefore to combine the above estimator of *P*(*wt*+1|*wt*−1,*wt*) with one obtained from a shorter suffix of the currently observed sequence. For example, here we can also predict the probability of *wt*+1 (given the context that precedes it) by dividing the number of occurrences of *wt*, *wt*+1 by the number of occurrences of *wt* (this is called a *bigram*). Similarly, using only the relative frequency of *wt*+1, one obtains a unigram estimator. The three estimators can then be combined, either by choosing only one of them in a particular context (e.g., based the frequency counts of the subsequences), or by combining them (usually in a linear mixture). A large literature on techniques to *smooth* frequency counts of subsequences has given rise to a number of algorithms and variants.

### 2.8.2 Distributed Representations

The idea of distributed representation has been at the core of the revival of artificial neural network research in the early 1980's, best represented by the connectionist bringing together computer scientists, cognitive psychologists, physicists, neuroscientists, and others. The main proponent of this idea has been Geoffrey Hinton. An early discussion can be found in the Parallel Distributed Processing book (1986), a landmark of the connectionist approach.

The idea of distributed representations was introduced with reference to cognitive representations: a mental object can be represented efficiently (both in terms of number of bits and in terms of number of examples needed to generalize about it) by characterizing the object using many features, each of which can separately each be active or inactive. For example, with m binary features, one can describe up to 2m different objects. The idea is that the brain would be learning and using such representations because they help it generalize to new objects that are similar to known ones in many respects. A distributed representation is opposed to a local representation, in which only one neuron (or very few) is active at each time, i.e., as with grandmother cells. One can view n-gram models as a mostly local representation: only the units associated with the specific subsequences of the input sequence are turned on. Hence the number of units needed to capture the possible sequences of interest grows exponentially with sequence length.

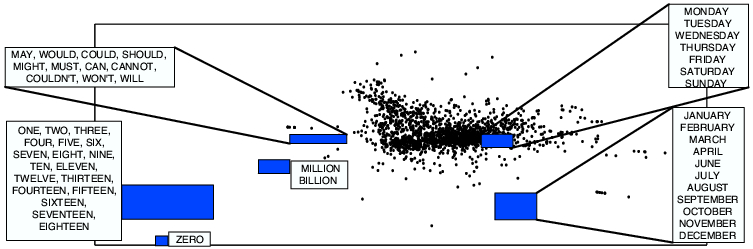


Figure 6: Example of 2-dimensional distributed representation for words obtained

Previously to the neural network language models introduced by Bengio, several neural network models had been proposed that exploited distributed representations for learning about symbolic data, modeling linguistic data and character sequences. It was demonstrated how distributed representations for symbols could be combined with neural network probability predictions in order to surpass standard n-gram models on statistical language modeling tasks. Experiments on related algorithms for learning distributed representations of words have shown that the learned features make sense linguistically. (Yoshua Bengio, n.d.)

# Related Work

In recent years, the rise of Massive Online Open Courses (MOOCs), and Technology Enhanced Learning (TEL) systems, in general, has highlighted even more the importance of having efficient and accurate information retrieval systems. This boom of online multimedia content has also brought an information overload problem. It is useless to have a large amount of online educational resources if students are not able to quickly find those that best suit their educational needs and preferences.

Information retrieval is a vast area of research with a large number of contributions in different domains (Baeza-Yates, Ribeiro, & others, 2011). Among them, many methods and algorithms have been proposed to find and retrieve textual and multimedia content from the web. In the latter case, multimedia IR encompasses different tasks, such as feature extraction and indexing from different types of sources (video frames, images, audio tracks, speech transcripts, etc.). There are several approaches proposed in the literature for multimedia IR: content-based IR; audio and music retrieval; speech recognition; or retrieving and browsing video.

In content-based multimedia IR, the primary objective is to identify and extract features related to image contents. Following this approach, in (Budnik, Gutierrez-Gomez, Safadi, & Quénot, 2015), authors compare ’traditional’ engineered (hand-crafted) features (or descriptors) and learned features for content-based semantic indexing of video documents. Learned (or semantic) features are obtained by training classifiers in the context of the TRECVID3 semantic indexing task.

In (Elleuch, Ammar, & Alimi, 2015), authors propose a combination of content-based video indexing approaches: text-based, feature-based, and semantic-based. The text-based approach focuses on using keywords or tags to describe video content. The feature-based approach aims to extract low-level features such as color, texture, shape and motion from the video data and use them as indexing keys. The semantic-based approach focuses on the automatic video content annotation with their semantic meanings.

Following an approach similar to the proposed in this paper, the work presented in (Iyer, et al., 2016) highlights the shortcomings of current multimedia indexing and retrieval techniques, mainly based on sparse tagging, and deals with content-based video indexing and retrieval using an LDA probabilistic framework (Blei, Ng, & Jordan, 2003).

In our same domain of lecture video retrieval, (Repp, Grob, & Meinel, 2008) presents an indexing method for recorded videos of computer science courses. This proposal uses the automatic transcriptions from a speech-recognition engine to create a chain index for detailed browsing inside a lecture video. Also, in (Yang & Meinel, 2014), authors presented a method for content-based video indexing and retrieval in sizable German lecture video archives. This paper applies a combination of automatic video segmentation and key-frame detection with a technique to extract textual meta-data by applying video Optical Character Recognition (OCR) technology on key-frames and Automatic Speech Recognition (ASR) on lecture audio tracks. Applying a technique based on multi-modal language models for lecture video retrieval, in (Chen, Cooper, Joshi, & Girod, 2014) authors demonstrated that this method outperforms LDA-based methods when speech transcripts are error-free, but the model shows similar performance for noisy text.

Other works use BoW based methods to classify and retrieve videos (with keywords (Basu, Yu, Singh, & Zimmermann, 2016), subjects (Van Nguyen, Coustaty, & Ogier, 2014) or visual features (Ngo, et al., 2007)). However, BOW based models cannot describe the content of an image objectively and neglect the spatial distribution of visual words and the order of words in transcripts.

Although multimedia retrieval based on speech transcripts may seem very similar to text retrieval, in practice, it is very different. For instance, speech transcripts lack from or have many flaws detecting structure (punctuation, paragraphs). Therefore, the appropriate proposals for feature extraction and indexing of written text are not always the best to perform this task on speech transcripts. The same happens with the approaches that use textual metadata or OCR to analyze lecture slides and get keywords from them. For our system, we take as input Spanish speech transcripts from an extensive database of lecture recordings from many types of university courses. Therefore, we focus on the problem of feature extraction and video indexing from speech transcripts (which may result in noisy text gathered from automatic speech recognition engines) to deliver the most suitable set of videos for a specific keyword search.

# System Design and Implementation

Searching within a large available set of videos represents a challenging task for a student. The current video search mechanism implemented in the multimedia platform <https://media.upv.es/> performs a full-text search over the title and keyword fields of the videos. It examines the words stored in such fields and tries to match with the search query made by the user. These full-search techniques may suffer from the lack of semantics and context on the search since they only take into account the word included in the query as it is. In the approach proposed in this work, we present a procedure that is much more context-aware, being able to classify videos according to their content using their transcripts.

## Outline of Data Analysis Pipeline

Figure 7: Data Analysis Pipeline

### Build the bag-of-words

As the video transcripts represent the primary input for the data analysis process, the first step is to build a BoW by tokenizing the transcripts to remove stop-words and words that are too short. Once the BoW has been created the next task is to determine the domains in which the transcripts may group. As no labels are being provided, the most effective solution is to implement an unsupervised learning algorithm (i.e., clustering) for grouping the transcripts.

The most significant limitation of this approach is represented by the fact that the BoW cannot be used directly as an input to a clustering algorithm and why we have to embed the transcripts using state of the art feed-forward neural net language models. Another issue may regard the fact that we are dealing with a large number of words from BoW and with a reasonably large number of transcripts, which will end up in having a sufficiently large input dataset for clustering. From the application domain perspective, a cluster should group transcripts that belong to a particular domain. By observing the university’s media website, we have identified three domains: Engineering & Architecture, Sciences (Biological Sciences), and Social & Legal.



### 4.1.2 Compute matrix with NNLM Word Embedding Results

We want to divide the transcripts into four different domains and to do that we have to run a clustering algorithm on the transcripts, but since we can’t run K-means on string objects, we have to transform our transcripts into a data format that our clustering algorithm understands. To accomplish this, we use the NNLM word embeddings. NNLM word embedding saves a lot of space by learning a distributed representation for words which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences.

The model learns simultaneously a distributed representation for each word along with the probability function for word sequences, expressed in terms of these representations. The generalisation is obtained because a sequence of words that have never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence (Bengio, Ducharme, Vincent, & Jauvin, 2003). The output from the NNLM embedding is a vector of 128 float numbers for each transcript, which amounts in total to a float matrix that is 16.012 lines (one for each transcript) and 128 columns. The output vector forms the input for our clustering algorithm.

The following code snipped shows how the NNLM model is accessed from the TensorFlow library and how the documents are embedded:



### 4.1.3 Build clusters of transcripts

The next step consists of running a clustering method for assigning items (i.e., transcripts) into groups. From this perspective, each obtained cluster represents a domain within the entire set of available videos. The primary purpose of the clustering algorithm is to bring together items (i.e., transcripts) for which the terms have similar NNLM embedding results over the entire transcripts corpus. One option is to use a standard simple k-means clustering algorithm (MacQueen & others, 1967) and provide a value for K as a domain knowledge person provides or use other algorithms that do not require for the number of clusters, such as xMeans (Pelleg, Moore, & others, 2000).

Finding the optimal number of clusters in a particular dataset represents itself a challenging research issue and is not covered by the current works. Another parameter that needs setup within the clustering process is the distance function (i.e., Euclidean, Jaccard, cosine, edit, etc.). Taking into account that the embedding results represent the input data, the current approach uses the Euclidean distance along with standard clustering quality metrics: SSE, homogeneity, completeness, Adjusted Rand-Index or Silhouette coefficient.



### 4.1.4 Model each cluster and find its topics and scores

For each cluster of transcripts (i.e., BoW corpus of that particular cluster and the dictionary of that cluster) we use LDA to determine its topics and associated scores. From an LDA perspective, the words are represented by the tokens obtained in the first step, the documents are represented by the transcripts and the corpus is represented by the set of transcripts for a particular cluster. As output, LDA determines the latent topics and their characterization in terms of words. Intuitively, for each cluster (i.e., domain) the LDA model creates a list of topics defined by scores whose values add up to one.

More, each topic is defined by a list of words with their coefficients. Both in the case of the topic’s scores and coefficient’s values, the interpretation is that a more significant number represents a more important topic or word. Finally, the model (i.e., the topic’s scores and their list of coefficients and words) is serialized for later querying.

The following code snippet shows how the dictionaries required for the LDA models are created and then how the LDA models themselves are created.

There is also the function that will return the Topics of an LDA model which are represented by the words that form the topic and their scores for that topic.

See *Table 2* for an example of the output generated by the function mentioned above.



### 4.1.5 Query and retrieve a ranked list of results

Once a query is obtained from a user, we need to find the cluster/domain to which it belongs. Since the query is regarded as a transcript, it is firstly preprocessed and its cluster is being determined. Determining the cluster to which the query belongs needs computing the NNLM embedding results of the words that make up the query. The embedding values for the query are being computed by considering the query as a transcript.

Once the embedding results from the query are determined, the closest centroid of already built clusters determines the cluster to which the query is assigned and therefore the LDA model to be queried. For each transcript from the assigned cluster, we compute the difference in topics between the query’s LDA results and transcript’s LDA results. A lower score indicates a smaller difference and therefore a transcript that matches the query better. Thus, once LDA provides the topics and scores for each cluster/domain, we may end up obtaining a ranked list of transcripts (i.e., movies) which are most similar to the query. We then take the top 5 transcripts and recommend those.

The following code snipped represents the function that returns the top 5 recommended video IDs that belong to the same cluster as our query and whose LDA scores are the closest to our query’s LDA scores.

See *Figure 8* and *Figure 9* for an example of how the output from the function looks like when displayed in the GUI and check *Table 3* from the Experimental Results section to see some examples of queries and their respective results.

Further analysis can be seen in *Table 4* where we can see the TextRazor results generated from the resulting top 3 transcripts.



## Graphical User Interface

The user enters a query in the box on the left and presses the ‘**Search**’ button to have the ranked list with the ID’s and the scores for the most relevant transcripts in the box below.

The graphical user interface is shown in the following figures:

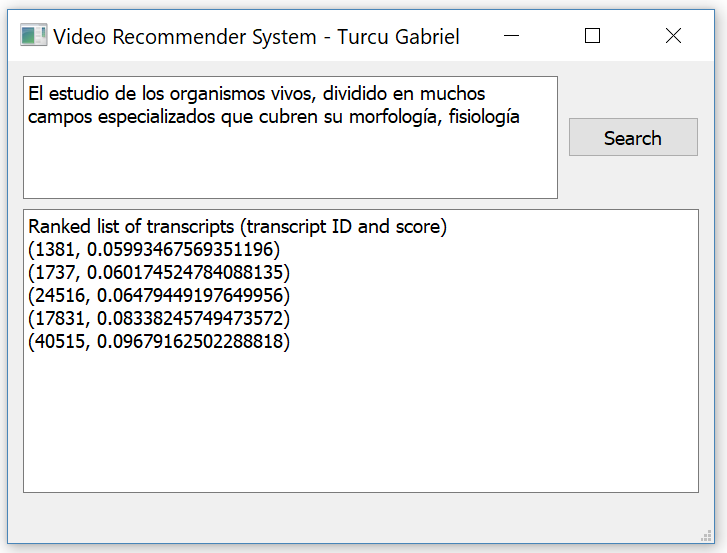


Figure 8: GUI Example 1

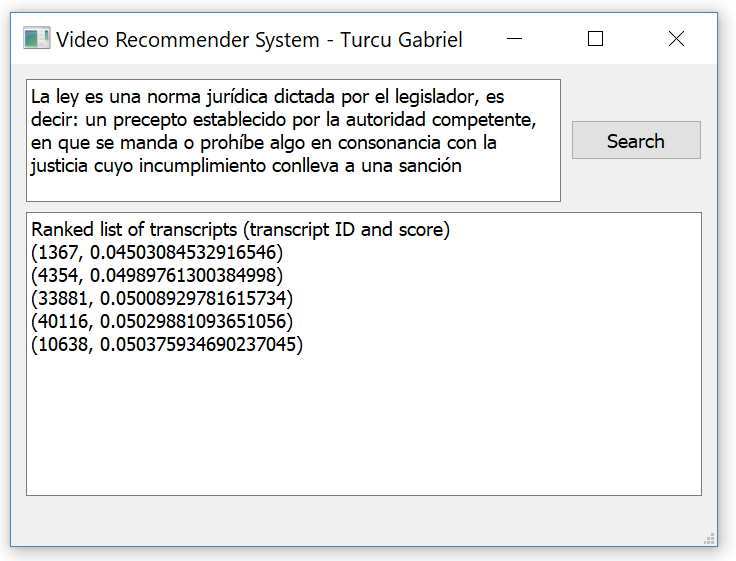


Figure 9: GUI Example 2

The following code snipped shows the GUI initialization and the function that gets called when the “Search” button from the GUI is pressed.



# Experimental Results

## Input Dataset

The input dataset consists of 16.012 video transcripts that are accessible through a json file. The structure and raw data for a record from the json file are presented in the following example.



This dataset comes from an online e-learning platform (i.e., <https://media.upv.es> ) from Universitat Politecnica de Valencia (UPV). The UPV’s platform has recording facilities to create educational videos (most of them in Spanish) which are finally publicly available as MOOCs.

The dataset includes the information necessary to process the video transcription and also includes additional information such as the identifiers needed to download the video (such as the **\_id field**, the **src** field or the **type** field). Similarly, the fields **mimetype, width and height** are the metainformation of the video file. Information such as the title of the video or its duration in seconds is also available.

However, something that characterizes these videos, which are educational videos recorded in a room dedicated to this activity, is that they are accompanied by information related to the slide presentation if used during the recording of the video. During the recording of the videos a software that detects the change of slide in the screen of the presenter was used. Thus, each slide detected in the dataset was captured and tagged. Therefore, the slides field contains a list of identified slides with their filename to be downloaded, the mime-type of the file and the instant (in seconds) of the video in which the slide change was detected.

What makes this dataset interesting is that it not only includes a collection of videos with a specific theme (educational videos), but also includes the transcription of the audio of these videos, and even each of the slides that have been used in the presentation, along with its temporary label. This approach enables a much more in-depth analysis of this data set.

## 5.2 Numerical Results

Each transcript from the raw input dataset has been transformed into a BoW for which NNLM values were computed and saved into a matrix. The dimension of the matrix is 16.012 (i.e., number of transcripts) x 128. What the embedding algorithm does is it maps from text to 128-dimensional embedding vectors.

Then, the embedding matrix has been used as input for simple k-means algorithms from Scikit-learn (Pedregosa, et al., 2011) to obtain a distribution of items (i.e., transcripts) into four clusters. For the clustering process, the items are represented by transcripts, features are represented words, and embedding results represent the values that build up the input dataset. *Table 1* presents several BoW samples along with computed embedding results and their corresponding CA (cluster assignments).

The running of simple K-means algorithm provides following distribution of transcripts into clusters: **22.71%**, **42.24%** and **35.04%**. Once we have determined the four clusters of transcripts, we further run the LDA algorithm to determine a lexical model of each cluster as a list of topics along with their scores and a list of words that make up that topic. The obtained model also consists of computed scores for each word within the topic. *Table 2* presents sample numerical results for the transcripts described in *Table 1*.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Sample BoW | Sample Embedding Results | CA |
| 3 | hola vamos a ver la parte cinco de la documentacion del software basicamente seria otro programa que nos quedaba veamos por explicar seria el cloc... | 0.8655922, 0.77715206, -0.16605175, -1.6408461, -0.8250744, 0.02193201, 0.700683, 0.7213742, -0.3716459, 1.5170424, 0.13685325, -0.20364942, 0.5022141, ... | 1 |
| 4 | hola vamos a ver ahora la parte politica de calidad es este digamos que el objeto que se realizaria simplifica de tareas basicamente estos se utiliza mucho... | 0.65827113, 0.5848848, 0.12338758, -1.5416939, -0.59042096, -0.17713968, 0.78873384, 0.5304664, -0.37286106, 1.0820895, 0.5255295, -0.27624443, 0.50664973, 0.15184559, ... | 1 |
| 6 | bienvenidos y bienvenidas a esta unidad de formacion en la cual trataremos sobre los usos de la letra cursiva somos sepalo assange del servicio de promocion y normalizacion... | 0.33565775, 0.48877674, 0.17549452, -1.6629573, -0.91474277, 0.3601234, 0.68089503, 0.55267304, -0.40141803, 1.3872166, 0.2669923, -0.25762805, 0.7387372, 0.17457546, ... | 3 |

Table 1: Sample BoW with embedding results and cluster assignments.

For each transcript presented in table 2 the computed topic scores have the sum equal to 1. This approach makes interpretation straightforward in the way that a topic score of 0.844 is a big score indicating that the topic represented is an excellent representative for that transcript. Further, each topic is represented by a list of words and their coefficients. In the same line of approach, larger value in coefficient is an indication of a more critical word among the words that make up the topic.

Once the query is being obtained from the user, it is preprocessed and embedding results are computed. At this stage, the corpus is represented by the available transcripts along with the query. Thus, the query is reduced to an array of words (i.e., a BoW corpus) which is assigned to the closest cluster (i.e., nearest centroid). Determining the cluster (i.e., the domain) to which the query belongs opens the way to further investigating its associated LDA model.

Table 3 presents the query results: the query, the assigned cluster (i.e., the domain), the LDA scores (i.e., the topic IDs and its score) and the ranked results (transcript ID and score). The results are ranked by the computed score from the fourth column as this score represents the difference between the query’s LDA score and transcript’s LDA score. A lower value in the transcript’s LDA score represents a smaller difference; therefore, a transcript that is a better match for the query.

|  |  |
| --- | --- |
| ID | LDA results: topics and scores |
| 3 | **Score**: 0.8441817164421082 **Topic**: 0.005\*"filtr" + 0.004\*"estad" + 0.002\*"clas" + 0.002\*"tension" + 0.002\*"senal"  **Score**: 0.0837591215968132 **Topic**: 0.002\*"dat" + 0.002\*"registr" + 0.001\*"formulari" + 0.001\*"eolic" + 0.001\*"electr"  **Score**: 0.06878719478845596 **Topic**: 0.001\*"datagr" + 0.001\*"estil" + 0.001\*"motor" + 0.001\*"dataset" + 0.001\*"wrait" |
| 4 | **Score**: 0.6999539732933044 **Topic**: 0.005\*"filtr" + 0.004\*"estad" + 0.002\*"clas" + 0.002\*"tension" + 0.002\*"seÃsal" ´  **Score**:0.24506276845932007 **Topic**:0.002\*"dat" + 0.002\*"registr" + 0.001\*"formulari" + 0.001\*"eolic" + 0.001\*"electr"  **Score**: 0.053080759942531586 **Topic**: 0.001\*"datagr" + 0.001\*"estil" + 0.001\*"motor" + 0.001\*"dataset" + 0.001\*"wrait" |
| 6 | **Score**: 0.6111074686050415 **Topic**: 0.001\*"oracion" + 0.001\*"pronombr" + 0.001\*"agu" + 0.001\*"sistem" + 0.001\*"instal"  **Score**: 0.22719042003154755 **Topic**: 0.001\*"plan" + 0.001\*"edifici" + 0.001\*"sistem" + 0.001\*"element" + 0.001\*"derech"  **Score**: 0.049981363117694855 **Topic**: 0.001\*"plan" + 0.001\*"control" + 0.001\*"derech" + 0.001\*"punt" + 0.001\*"nod"  **Score**: 0.04900844022631645 **Topic**: 0.001\*"fibr" + 0.001\*"atom" + 0.001\*"molecul" + 0.001\*"temperatur" + 0.001\*"sistem"  **Score**: 0.04536473751068115 **Topic**: 0.001\*"arrend" + 0.001\*"derech" + 0.001\*"pacient" + 0.001\*"valencian" + 0.001\*"anten"  **Score**: 0.010526408441364765 **Topic**: 0.001\*"sistem" + 0.001\*"pec" + 0.001\*"ulcer" + 0.001\*"presion" + 0.001\*"aliment" |

Table 2: Sample LDA models with their topics and scores

|  |  |  |  |
| --- | --- | --- | --- |
| Query | CA | LDA scores (topic ID and score) | Ranked list of transcripts (transcript ID and score) |
| Ciencias de la Computacion | 2 | (0,0.4422385) (1,0.053041767) (2,0.04431011) (6,0.033507172) (7,0.42583835) | (41107,0.10977402180433274) (765,0.11200470328330994) (18460,0.11249668002128602) (3895,0.11415494084358216) (1236,0.11463153958320618) |
| aprendizaje automatico | 2 | (9,0.6999294) (8,0.033338577) (7,0.03334638) | (29906,3.75695526599e-05) (16256,4.62792813777e-05) (35066,5.43296337155e-05) (3212,6.037577986724e-05) (17793,0.0001398846507487) |
| permanente para la proteccion de los animales en cria instituido | 2 | (6,0.014290362) (7,0.8714059) (8,0.014287368) (9,0.014287801) | (44794,1.2902542948724e-05) (27277,1.7145648598676e-05) (44466,0.03214275650680065) (26721,0.04286431334912777) (41793,0.04287060722708702) |

Table 3: Query results.

## Validation

Validation is the part of the project that has proved to be the most difficult mostly because we are working with real-life data and therefore, we do not possess the ground truth needed so that we may validate our work with ease. For validation, a couple of approaches have been investigated and tested including:

* Manual validation by looking at the results from a query which represent the top transcripts whose LDA scores are the closest to our query’s LDA scores and seeing if the content of the transcripts reflects the query.
* Applying TextRazor’s state-of-the-art Natural Language Processing and Artificial Intelligence API (Crayston) to parse, analyze and extract semantic meta-data from the transcripts to see if the detected Categories and Topics are related to our query. Table 4 shows the TextRazor generated results for the recommended transcripts associated with some queries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Query | Ranked list of transcripts | TextRazor Output for the first transcripts | | |
| First | Second | Third |
| La energía nuclear o atómica es la que se libera espontánea o artificialmente en las reacciones nucleares | (26798, 0.0575)  (40198, 0.0583)  (39614, 0.0589)  (30846, 0.0670)  (28567, 0.0829) | 0.54 - science and technology  0.45 - environment  0.42 - science and technology>mechanical engineering  0.42 - environment>natural resources>water  0.36 - science and technology>technology and engineering>civil engineering | 0.69 - science and technology>natural science>physics  0.48 - science and technology>natural science>chemistry  0.44 - science and technology>natural science  0.43 - science and technology>mechanical engineering  0.41 - science and technology>natural science>physics>nuclear physics>particle physics  0.41 - science and technology>natural science>physics>nuclear physics | 0.62- science and technology>natural science>chemistry  0.55 - science and technology  0.55 -economy, business and finance>economic sector>chemicals  0.50 - science and technology>natural science>physics  0.45 - science and technology>natural science  0.38 - science and technology>natural science>physics>nuclear physics>particle physics |
| La ley (en latín, lex, legis) es una norma jurídica dictada por el legislador, es decir: un precepto establecido por la autoridad competente, en que se manda o prohíbe algo en consonancia con la justicia cuyo incumplimiento conlleva a una sanción | (32529, 0.0244)  (38848, 0.0248)  (1509, 0.0250)  (44365, 0.0250)  (22254, 0.0306) | 0.64 - politics>government  0.64 -crime, law and justice>law  0.62 -crime, law and justice>law>civil law  0.59 - society>values>ethics  0.57 - politics  0.53 - politics>government>constitution (law)  0.52 - science and technology  0.46 - crime, law and justice>law>civil law>regulation  0.46 - society>values  0.46 - society | 0.70 - crime, law and justice>law  0.68 - politics>government  0.67 - crime, law and justice>law>civil law>regulation  0.64 - politics  0.62 - crime, law and justice>law>civil law  0.53 - politics>government>constitution (law)  0.48 - politics>government>heads of state  0.48 - society>values>ethics  0.46 - society  0.43 - crime, law and justice | 0.59 - economy, business and finance>economy  0.59 - science and technology>social sciences>economics  0.53 - crime, law and justice>law  0.52 - politics  0.46 - economy, business and finance>economic sector  0.45 - science and technology>social sciences  0.45 - politics>government |

Table 4: TextRazor generated results for some queries

* We applied the LSTM Siamese Text Similarity (Srivastava) algorithm on the top 3 results from a couple of queries to check if the recommended transcripts for those queries are similar to each other. We had to use the English transcripts because there wasn’t anything already trained for the Spanish language. We downloaded the transcripts that were translated from Spanish to English and applied the algorithm. If they would have been marked as similar by the LSTM Siamese algorithm and if the categories and topics from TextRazor would be relatable to our queries, that would tell us that our program does a good job at recommending transcripts based on the content in them. The results from the LSTM Siamese algorithm were inconclusive, however. This happened because the LSTM Siamese algorithm works best if the sentences are short and if they retain a meaning while our transcripts were detected imperfectly by the speech recognition engine after which they were imperfectly translated into English and there was also the issue that the transcripts are very long with the average number of words in a transcript being a bit over 1000 words.

# Conclusions and Future Works

In this paper, we have implemented a custom procedure for indexing and retrieving video transcripts. The input transcripts are from educational videos available from <https://media.upv.es/>

The data analysis pipeline consists of tokenizing, computing NNLM embedding results, clustering transcripts and building one LDA model for each transcript. Once the LDA models are available, the query provided by the user is preprocessed and labelled (i.e., assigned to a cluster) and the corresponding LDA model is used for obtaining the most similar topics (i.e., with highest scores) and therefore the most significant words with their coefficients. These results trace back to the originating transcripts, and consequently, a ranked list of videos is obtained as output. One of the critical limitations of the current approach is that it uses a fixed number (i.e., three) of clusters/domains.

This approach is due to practical reasons since we do not have any ground truth indicating the exact number of clusters that are in the 16.012 transcripts. Finding the correct number of clusters from the dataset may bring significant improvements regarding the relevance of the obtained ranked list. Besides, a future goal is to improve the current search mechanism which takes into consideration only the title of the video. Another shortcoming is that no ground-truth data is available and no mechanism to monitor learner’s search queries or search truth judgments. Coping with these shortcomings may end up in building a recommender system that takes into account the context of the learner and previously trained classifiers.

# **7 References**

* Baeza-Yates, R., Ribeiro, B. d., & others. (2011). *Modern information retrieval.* New York: ACM Press; Harlow, England: Addison-Wesley,.
* Bakar, Z. A., Kassim, M., Sahroni, M. N., & Anuar, N. (2017). A Survey: Framework to Develop Retrieval Algorithms of Indexing Techniques on Learning Material. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9*, 43-46.
* Basu, S., Yu, Y., Singh, V. K., & Zimmermann, R. (2016). Videopedia: Lecture Video Recommendation for Educational Blogs Using Topic Modeling. *International Conference on Multimedia Modeling*, (pp. 238-250).
* Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research, 3*, 1137-1155.
* Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research, 3*, 993-1022.
* Budnik, M., Gutierrez-Gomez, E.-L., Safadi, B., & Quénot, G. (2015). Learned features versus engineered features for semantic video indexing. *Content-Based Multimedia Indexing (CBMI), 2015 13th International Workshop on*, (pp. 1-6).
* Chen, H., Cooper, M., Joshi, D., & Girod, B. (2014). Multi-modal language models for lecture video retrieval. *Proceedings of the 22nd ACM international conference on Multimedia*, (pp. 1081-1084).
* Cournapeau, D. (2019). *scikit-learn*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Scikit-learn
* Crayston, T. (n.d.). The Natural Language Processing API.
* Elleuch, N., Ammar, A. B., & Alimi, A. M. (2015). A generic framework for semantic video indexing based on visual concepts/contexts detection. *Multimedia Tools and Applications, 74*, 1397-1421.
* Google Brain Team. (n.d.). *TensorFlow*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/TensorFlow
* Iyer, R. R., Parekh, S., Mohandoss, V., Ramsurat, A., Raj, B., & Singh, R. (2016). Content-based video indexing and retrieval using corr-lda. *arXiv preprint arXiv:1602.08581*.
* *K-means clustering*. (2019). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/K-means\_clustering
* *Latent Dirichlet allocation*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Latent\_Dirichlet\_allocation
* MacQueen, J., & others. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, *1*, pp. 281-297.
* Ngo, C.-W., Jiang, Y.-G., Wei, X.-Y., Wang, F., Zhao, W., Tan, H.-K., & Wu, X. (2007). Experimenting VIREO-374: Bag-of-Visual-Words and Visual-Based Ontology for Semantic Video Indexing and search. *TRECVID.*
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . others. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research, 12*, 2825-2830.
* Pelleg, D., Moore, A. W., & others. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. *Icml*, *1*, pp. 727-734.
* *Python (programming language)*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Python\_(programming\_language)
* Řehůřek, R. (2019). *Gensim*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Gensim
* Repp, S., Grob, A., & Meinel, C. (2008). Browsing within lecture videos based on the chain index of speech transcription. *IEEE Transactions on learning technologies, 1*, 145-156.
* Sebastian Thrun, Katie Malone. (2013). *Intro to Machine Learning*. Retrieved from Udacity: https://eu.udacity.com/course/intro-to-machine-learning--ud120
* Srivastava, A. (n.d.). LSTM Siamese Text Similarity.
* TANNER, G. (2018). *What is Machine Learning and why is it important*. Retrieved from gilberttanner.com: https://gilberttanner.com/2018/09/04/what-is-machine-learning-and-why-is-it-important/
* Van Nguyen, N., Coustaty, M., & Ogier, J.-M. (2014). Multi-modal and cross-modal for lecture videos retrieval. *Pattern Recognition (ICPR), 2014 22nd International Conference on*, (pp. 2667-2672).
* Yang, H., & Meinel, C. (2014). Content based lecture video retrieval using speech and video text information. *IEEE Transactions on Learning Technologies*, 142-154.
* Yoshua Bengio. (n.d.). *Neural net language models*. Retrieved from Scholarpedia: http://www.scholarpedia.org/article/Neural\_net\_language\_models
* Yousef, A. M., Chatti, M. A., & Schroeder, U. (2014). Video-based learning: a critical analysis of the research published in 2003-2013 and future visions. eLmL.