BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

TOPIC IDENTIFICATION FROM SPOKEN TED-TALKS

IDENTIFIKÁCIA TÉM Z HOVORENÝCH TED-TALKS

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ADAM VAŠŠ**
AUTOR PRÁCE

SUPERVISOR **SANTOSH KESIRAJU**
VEDOUCÍ PRÁCE

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) — Academic year 2018/2019

Bachelor's Thesis Specification

22509

Student:	**Vašš Adam**
Programme:	Information Technology
Title:	**Topic Identification from Spoken TED-Talks**
Category:	Speech and Natural Language Processing

Assignment:

1. Get acquainted with basic building blocks of automatic speech recognition and also classifiers in field of machine learning.
2. Conduct a literature survey on topic identification from spoken audio.
3. Build an automatic speech recognition system with the help of publicly available tools (eg: Kaldi) and transcribe the spoken TED-talks into text.
4. Considering the tags available for TED-talks, build a multi-label data set with balanced label proportions.
5. Train and test topic identification classifiers (at least 2) on the created data set and produce benchmarking results.
6. Create a poster describing your work.

Recommended literature:

- T. J. Hazen, F. Richardson and A. Margolis, "Topic identification from audio recordings using word and phone recognition lattices," *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Kyoto, 2007, pp. 659-664.
- A. Rousseau, P. Deleglise, Y. Estve, "TED-LIUM: an Automatic Speech Recognition dedicated corpus", *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012),* 2012.

Requirements for the first semester:

- Points 1-3

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor:	**Kesiraju Santosh**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	July 1, 2019
Submission deadline:	July 31, 2019
Approval date:	July 12, 2019

Abstract

This thesis deals with the problems of language recognition and topic classification, using TED-LIUM corpus to train both the ASR and classification models. The ASR system is built using the Kaldi toolkit, achieving the WER of 16.6%. The classification problem is addressed using linear classification methods, specifically Multinomial Naive Bayes and Linear Support Vector Machines, the latter method achieving higher topic classification accuracy.

Abstrakt

Táto práca sa zaoberá problémom spracovania prirodzeného jazyka a následnej klasifikácie. Použité systémy boli modelované na TED-LIUM korpuse. Systém automatického spracovania jazyka bol modelovaný s použitím sady nástrojov Kaldi. Vo výsledku bol dosiahnutý WER s hodnotou 16.6%. Problém klasifikácie textu bol adresovaný s pomocou metód na lineárnu klasifikáciu, konkrétne Multinomial Naive Bayes a Linear Support Vector Machines, kde druhá technika dosiahla vyššiu presnosť klasifikácie.

Keywords

TED, talks, topic identification, machine learning, classification, transcription, linear classification, Kaldi, support vector machines, acoustic modeling, language modeling, TED-LIUM, ASR

Klíčová slova

TED, talks, identifikácia tém, strojové učenie, klasifikácia, transkripcia, lineárna klasifikácia, Kaldi, support vector machines, akustický model, lingvistický model, TED-LIUM, ASR

Reference

VAŠŠ, Adam. *Topic Identification from Spoken TED-Talks.* Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Santosh Kesiraju

Topic Identification from Spoken TED-Talks

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Mr. Santosh Kesiraju. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

. . . .
Adam Vašš
July 31, 2019

Acknowledgements

I wish to express my sincere thanks to Mr. Santosh Kesiraju for providing his expertise and guidance for this thesis. Thank you!

Contents

Chapter 1

Introduction

Speech is the most natural form of communication for humans. Machines, unfortunately, have never been very keen speakers — or listeners, for that matter, due to fundamental differences in information processing. While it is true that machines have been able to store and reproduce audio information in detail (that no human can hope to match) for a long time now, they have always been lacking the crucial capabilities of comprehension and understanding.

Most modern smartphones are able to record and reproduce spoken word, but when it comes to recognizing *what* it is that the sequence of bits representing audio signal is about, machines run into a seemingly insurmountable obstacle.

This thesis deals with machine recognition, specifically speech recognition, and subsequent classification. So that the machine using the proposed model can not only record whatever it is "hearing", it can also recognize what is the spoken document *about*.

In the first chapter (2), the general problem of speech recognition is introduced, together with the parts of modern automatic speech recognition systems, as well as the toolkit used for the purpose of building the speech recognition models in this paper.

Following that, there is an introduction to the second problem, specifically the topic identification (3). From there the more hands-on parts of this paper is described — the implementation details (4).

Observations of the outcome can be read in the last chapter, the conclusion (5).

Chapter 2

Speech recognition

This chapter talks about the first part of the TED-talk topic identification problem and that is the speech recognition and transcription. It describes the parts of the speech recognition model, the training approach, as well as the chosen toolkit.

Speech recognition is generally considered to be one of the difficult problems in computer science. There are numerous benefits, and therefore motivations, to solving the problem (or achieving a sufficient success rate for the specific domain). Acquiring the ability to use natural language as a user interface[1] leads to broad applications in many areas, whether that is voice control of personal or embedded computers, transcriptions, education (e.g. learning correct word pronunciations in new languages), improving accessibility for people with disabilities, etc.

These applications require rather robust ASR systems, that must be able to function reliably when facing non-ideal environmental conditions — real-world speech includes various acoustic distortions and noise. Even though ASR has been a field of research for over half a century, and despite the considerable progress in the recent decades, it still remains a challenge.

The very first developed ASR system was named Audrey. It was developed at Bell Labs in 1952 and was capable of recognizing spoken single digits (uttered exactly 350ms apart) with a success rate between 97% and 99% [5]. In order to do so, it located the formants[2] in the power spectrums of the individual utterances. This system, like many that came afterwards, had its success rate heavily dependent on a number of constraints - dependence on a single speaker, isolated words, small vocabulary and a constrained grammar. As ASR systems evolved, the number of constraints had been gradually reduced, as well as the impact of the ones still outstanding. This development eventually led to systems capable of recognizing more complex (grammar and vocabulary-wise) speech, free of the single speaker dependencies (respectively able to dynamically calibrate themselves to a specific speaker). The speaker dependence problem had been considered as the one that was the most difficult to overcome, since most parametric representations of speech are highly speaker dependent — one set of parameters can perform well for one speaker, but poorly for another [13].

Most current ASR systems rely on a combination of Hidden Markov models (HMMs) and either Gaussian mixture models (GMMs), or deep neural networks (DNNs).

[1]Also known as LUI, or NLUI

[2]Spectral shaping that results from an acoustic resonance of the human vocal tract

2.1 Formal definition

General goal of the ASR[3] systems is to determine the most probable word sequence $\tilde{W}$ given the observed acoustic signal Y

$$\tilde{W} = argmax\ P(W|Y) = \frac{argmax\ P(W)P(Y|W)}{P(Y)} \tag{2.1}$$

ASR performs a search for the word sequence $\tilde{W}$ that maximizes $P(W)$ and $P(Y|W)$ where

- $P(W)$ is the language model, i.e. the likelihood of the word sequence
- $P(Y|W)$ is the acoustic model, i.e. the likelihood of the observed acoustic signal, given word sequence

Equation 2.1, and figure 2.1 in this section are adopted from [7].

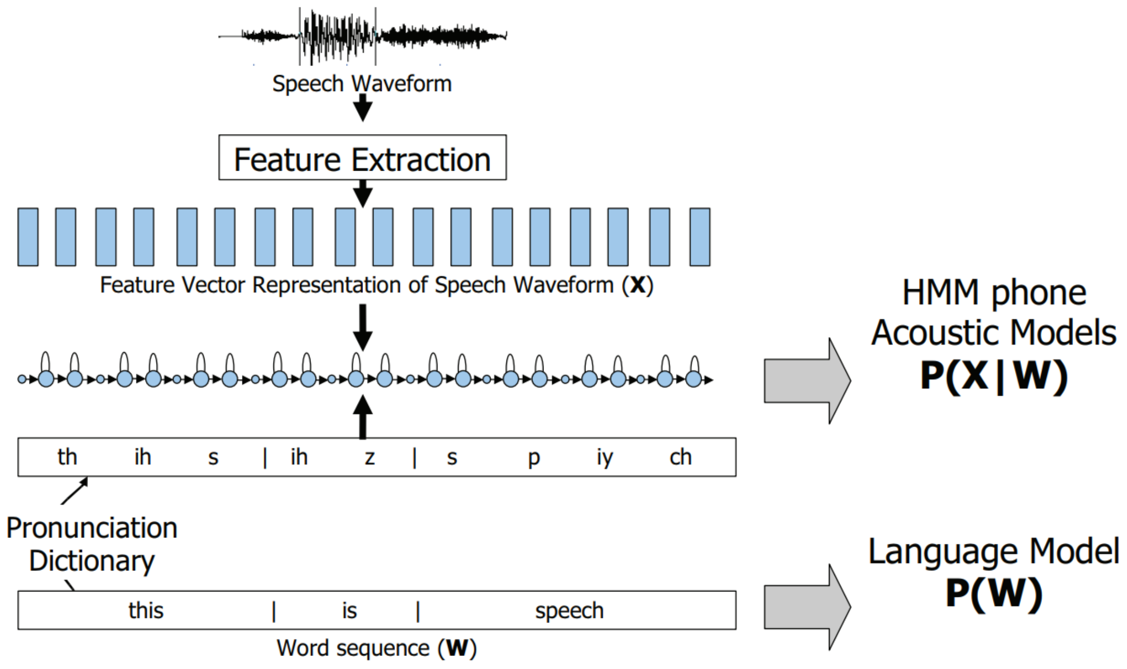

Figure 2.1: Parts of an ASR system

2.2 Building an ASR system

The process of building an ASR system generally consists of three major steps — feature extraction, acoustic modelling and language modelling. There are other, optional, steps, such as chunking (splitting the training data into smaller parts). Chunking makes it possible to make use of parallel processing to speed up the process of training the model. The extent to which this approach hastens the model's training depends on the used data set and the underlying hardware.

[3]Automatic speech recognition

2.2.1 Word error rate

Performance metric of choice for the ASR system used in this paper is the word error rate. While there are many other potential metrics to be considered, they either tend to be difficult to quantify and compare, or have their significance diminished by the nature of the explored problem, which is producing transcripts to serve as an input for the topic classifier. Metrics such as training or recognition time, bandwidth, etc. are secondary to the model's accuracy in this case. Considering further applications, for example a real-time speech topic identifier, those would become increasingly important, as other factors would come into play (such as user experience).

Word error rate (WER) is derived from the Levenshtein distance [14], and can be expressed as follows;

$$WER = \frac{S + D + I}{N} \tag{2.2}$$

[2] where

- S - substitutions (changing a word)
- D - deletions (missing a word)
- I - insertions (adding an extra word)
- N - word count (words said in total)

Individual types of errors can be weighted differently, for example penalizing substitutions more severely than deletions, or vice versa.

2.3 Feature extraction

Feature extraction is an essential part of signal pre-processing, intended to select relevant attributes (features) from the data set. This leads to data reduction, as well as performance improvement, considering both computational complexity and the effectiveness of the resulting model — provided that the right features are constructed and selected, otherwise this step may be detrimental to the model's effectiveness [6].

Feature can be understood as an attribute of raw data. Selecting the right features, as well as the right extraction method, is a domain specific problem and will inherently vary between applications. The main goal is to extract the most information-rich elements, while omitting the rest.

Feature extraction is composed of two steps:

1. *Feature construction* - by the means of standardization, normalization, signal enhancement, etc.
2. *Feature selection* - using filters, wrappers or embedded methods.

In other words, the goal of feature extraction is, to first define, and subsequently select, the subset of data that represents the observed patterns most effectively (considering the ratio of raw data and relevant information encompassed within).

The toolkit of choice for building the ASR system used for this paper, Kaldi, supports creation (and selection) of Mel-frequency cepstral coefficients (MFCC) as well as perceptual linear predictive (PLP) features. The built model on MFCC features, which is one of the most common techniques used within modern ASR systems. MFCCs are based on frequency domain and are usually considerably more accurate than their time domain based counterparts. MFCCs use the Mel scale, which is modelled on the human ear scale [4].

2.4 Acoustic Model

One of the integral parts of an ASR system is the acoustic model, which facilitates the mapping between the speech features extracted from audio signal and the respective linguistic units (e.g. phones). In order to prepare the acoustic model, both audio files and the transcripts are necessary - in this case both are included in the TED-LIUM corpus.

HMM-GMM ASR systems rely on using hidden Markov models to address the temporal variability of speech, and Gaussian mixture models to determine the quality of the fit between every state of every HMM and the acoustic input represented as a frame of coefficients [10].

Kaldi toolkit prepares the acoustic model operating on the phoneme-level (as opposed to the word-level) [8].

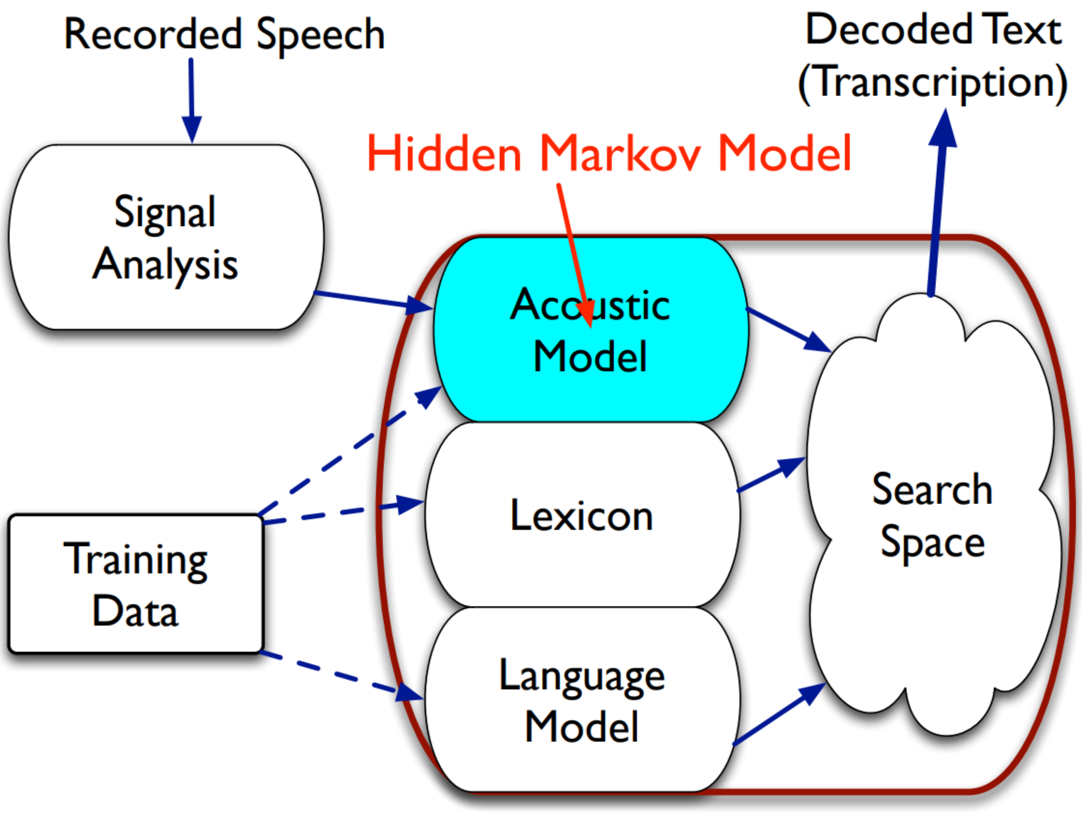

Figure 2.2: Acoustic model [23]

2.4.1 Markov model

The ASR model built for the purpose of this paper relies on Hidden Markov models. However in order to understand the way they work, it is necessary to understand what Markov models are and what is their application.

Observations of real-life processes can be represented as signals - either discrete (like characters of a finite alphabet) or continuous (like speech samples). Their sources can be stationary (when their statistical properties do not vary over time) or non-stationary. Aside from whether they are discrete or continuous, signals can also be pure (originating from a single source) or corrupted by other signal sources (noise), reverberations or distortions.

Acoustic and language modelling operates under Markov assumption — that is an assumption that the probability of future states depends only on the present state and not on the past states.

There are many practical applications, like recognition or identification systems, that require a way to characterize these signals in terms of signal models. [19]

2.4.2 Hidden Markov model

Hidden Markov model is a specific type of a Markov model, for which the system being modeled is a Markov process with states that are not observable. As such, Hidden Markov model contains an embedded stochastic process that is not observable and can only be observed by another stochastic process.

2.5 Language Model

Statistical language modelling is another integral part of the ASR system. Language model is essentially a probability distribution over strings in a finite alphabet. There are many techniques for estimating the probabilities using the knowledge available about the language generation process of the language in question.

For example, let's consider a binary alphabet consisting of two symbols - 0 and 1, with a known generation mechanism defined as

$$\begin{aligned}
P(x_i = 0|x_{i-1} = 0) = 0.9 \\
P(x_i = 0|x_{i-1} = 1) = 0.1 \\
P(x_i = 1|x_{i-1} = 1) = 0.9 \\
P(x_i = 1|x_{i-1} = 0) = 0.1
\end{aligned} \tag{2.3}$$

This combination of probabilities can be thought of as a language model for the language in question. Relying on a language model is advantageous because the way it is assembled (i.e. using collection statistics) is transparent, and the user does not need to rely on heuristics to understand some of the more obscure processes [17].

As can be observed in equation 2.3, what language models do is that they assign *probabilities* of the next possible word, or a sequence of words. Depending on their application, the language models may be dealing with a single next word at a time (unigram), or n-grams of words, or even entire sentences.

When applied to words, it is necessary to discern between the depth in word history, that is, how long are the word sequences (n-grams) that are being considered. The probability of a word sequence W for 1-word history would be

$$P(W) \approx P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdots P(w_N|w_N - 1) \tag{2.4}$$

building on that, 2-word history probability could be represented like so

$$P(W) \approx P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_N|w_N - 2, w_N - 1) \tag{2.5}$$

Since 2-word history describes the probabilities of three word sequences (two “historical” words and the one following them), it is represented as a trigram (3-gram) model. Probabilities $P(w_a|w_b, w_c)$ are estimated as

$$P(w_a|w_b, w_c) = \frac{C(w_b, w_c, w_a)}{C(w_b, w_c)} \tag{2.6}$$

$C(\cdot)$ represents the count of the word (or a sequence of words) in the respective data set.

In other words, the language model mimics our *prior* knowledge of the language that it is modeled upon — which word sequences of words are common, and which are less so. The data from the acoustic model (or acoustic score) serves to complement the language model in choosing the word sequences most likely matching the uttered sounds.

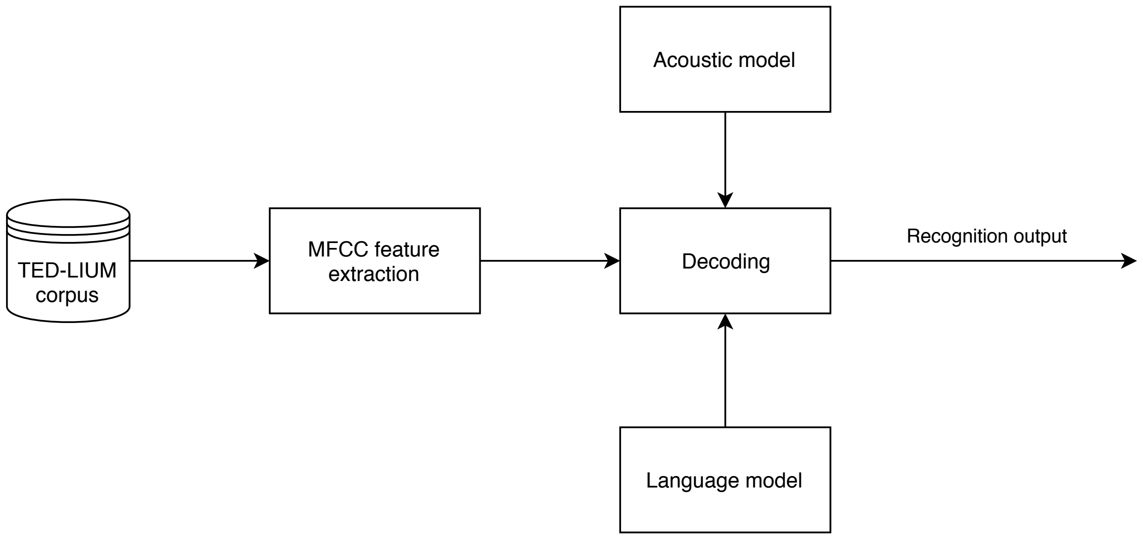

Figure 2.3: ASR system modeling process

2.6 Kaldi

Kaldi is a speech recognition toolkit written in C++ and licensed under the permissive and free Apache License v2.0. [18]. The speech recognition system implemented by Kaldi is based on finite-state transducers, which are built on the OpenFst library. Kaldi toolkit comes with prepared recipes for building speech recognition systems, built on various freely available corpora, while its extensible design allows the user to prepare their own “recipes” or modify the existing ones.

Kaldi uses an incremental approach to ASR system modelling, trying to improve the speech recognition accuracy over multiple passes. These passes involve re-scoring the lattices, recomputing the graphs, as well as decoding the training data using the updated models.

The first step is building a mono phone model, that is, a model only mapping the speech features onto monophones. Following the incremental logic described above, Kaldi builds a tri-phone model (sequence of three phones) which is then re-scored over multiple iterations, with the intention to improve the accuracy (reduce the word error rate, WER).

Iteration	Dev	Dev_rescore	Test	Test_rescore
1	27.5%	26.1%	27.2%	25.8%
2	22.9%	21.6%	22.1%	20.9%
3	19.0%	17.8%	17.5%	16.6%

Table 2.1: Word error rates' evolution over multiple training and scoring iterations

While the differences between singular steps may seem negligible at first, the small improvements stack up, as can be observed by the differences between the first and the last iterations.

The decoding graph consists of the following parts

- H contains the HMM 2.4.2 definitions - takes transition-ids as the input, producing context-dependent phones as the output
- C is the context-dependency - takes context-dependent phones as the input, producing phones
- L is the lexicon - takes phones as the input, producing words as the output
- G is an acceptor encoding the grammar or language model

For the purposes of this paper it is used to create the speech recognition model, based on the TED-LIUM corpus, second release 4.1.

Chapter 3

Topic identification

In this chapter we will go over the principles and methods of text processing and classification, which will be necessary in order to effectively identify the topics of the transcribed TED talks.

3.1 Data Representation

We need to assign each term a score, that denotes its importance within the document. The simplest approach to achieve this is by assigning the term's weight according to the number of occurrences in the document.

3.1.1 Bag-of-words model

In order to address the classification problem, it is necessary to first process the document... One of the methods to achieve this is known as the *bag-of-words model.* Using this model, we do not care about the ordering of the individual words, while counting the number of occurrences of each word. Opting for this way of document representation leads to some information loss (for example sentences using the same words in different order would end up being viewed as identical) — it is crucial to consider the usage before choosing the proper data representation approach.

3.1.2 N-gram model

While the bag-of-words model considers individual words independently, N-gram model works with tuples of words. Using this model allows the retention of relations between the terms, which would be lost when basing the weights purely on the term occurrences as in the bag-of-words representation.

3.1.3 Term frequency

The simplest measure for assigning term weights in text is assuming that the weight is directly proportional to the number of occurrences of the term in a document. The weight of a term t in a document d is defined as

$$W(d,t) = TF(d,t) \tag{3.1}$$

where $TF(d,t)$ stands for the term frequency of a term t in a document d [24].

3.1.4 Collection frequency

Collection frequency is the number of occurrences in the entire collection (or corpus) as opposed to a single document.

3.1.5 Document frequency

While the term frequency stands for the number of the occurrences of a term within a document, document frequency is a metric describing a collection of documents.

Document frequency is the number of documents containing the specific term.

3.1.6 Inverse document frequency

For the purposes of classification, the rarely occurring terms are often more important than their more common counterparts — therefore it is the inverse of the document frequency that is relevant for this application (the rarer the term is, the greater weight it should receive). Inverse document frequency is given by

$$IDF(t, D) = \log \tfrac{N}{df(t)} \tag{3.2}$$

N being the total number of documents in a collection and $DF(t)$ the number of texts that contain the term (document frequency).

Inverse document frequency denotes the uniqueness, or specificity, of the term, while term frequency describes its prevalence. By combining the two metrics, weighting precision will be improved

$$W(D, t) = TF(d, t) \cdot IDF(t) \tag{3.3}$$

as proposed in [22].

3.1.7 Text cleaning

Stop words

Another problem to consider is the level of importance of individual words. In the English language there are many terms commonly found in any text, for example:

- Pronouns: *I, me, they...*
- Articles: *a, the...*
- Verbs: *can, should, go...*

Due to their prevalence, they do not provide valuable information for the purpose of topic identification, but they do take up extra processing time and space. Considering this, it is preferred to omit these common words, commonly referred to as "stop words". There are many open stop word corpora available online, which can be used during the tokenization process to skip their indexing and omit their inclusion in the resulting data representation.

Special characters

Special characters are another possible subject to be cleaned from the text used in the classification pipeline.

3.2 Text classification

This sections briefly describes the most common classification methods, which were experimented with for the purpose of this paper.

Classification can be understood as the problem of determining which class does a given object belong to. This paper delves into the area of text classification — also referred to as topic classification or identification.

The need for text classification is hardly a novel need for society... (move to intro?)

Let us consider a description $d \in X$ of a document, where X is the document space, as well as a fixed set of classes $C = \{c_1, c_2, \cdots, c_N\}$. The classes can also be referred to as *categories* or *labels*. They are usually manually defined by a human. For the purpose of training a classification model, it is necessary to have a training set D with labeled documents $\langle d, c \rangle$, where $\langle d, c \rangle \in X \times C$.

For example

$$\langle d, c \rangle = \langle \text{Kaldi is a speech recognition toolkit, Technology} \rangle \tag{3.4}$$

In this case, d is a single-sentence document; "Kaldi is a speech recognition toolkit" and its label it "Technology". Using a learning algorithm, a classifier (or a classification function) γ mapping documents to classes can be obtained:

$$\gamma : X \longrightarrow C \tag{3.5}$$

This sort of learning is called *supervised*, since there is a human acting as an arbiter, or supervisor, defining the classes of the training data set.

3.2.1 Binary classification

The simplest form of classification, also known as binomial classification, deals with the problem of classifying individual data set elements into one of the two classes, based on some qualitative property.

There are many potential applications for binary classifiers, across many fields — such as when trying to determine whether an e-mail belongs to spam (technology), whether a tumor is malign or benign (medicine), whether a website visitor has a purchasing intent and thus make a good target for remarketing campaigns (business) and so on. [15]

3.2.2 Naive Bayes

Multinomial Naive Bayes, or multinomial NB model, is a robust probabilistic learning method. The probability of a document d being a representative of a class c is defined as

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{3.6}$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c. $P(c)$ stands for the "p"riorprobability of a document d being in a class c.

For the purposes of text classification, the goal is to retrieve the best class for the document, i.e. the class with the highest probability — "m"aximum a posteriori(MAP) class $c_m ap$

$$c_{map} = \underset{c \in C}{argmax}\ \hat{P}(c|d) = \underset{c \in C}{argmax}\ \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c) \tag{3.7}$$

P is denoted as $\hat{P}$ since the true values of the parameters $P(c)$ and $P(t_k|c)$ are unknown, the estimates are obtained from the training data set.

Using Naive Bayes model means operating under the assumption of conditional independence, or, in other words, an assumption that attribute values are independent of each other. That is also where the “naive” adjective comes from, as such assumption is often not matching reality very closely — this is especially true for ASR systems, as there are many interdependent terms in natural language.

Considering that, a question may arise whether this model should be at all considered for the problem of speech classification, since it relies on oversimplifying assumptions. However, even though the assumptions are limiting, its classification decisions are good. Due to the nature of the multinomial model from 3.7, the winning class tends to have significantly higher probability than the other contenders.

All things considered, Multinomial Naive Bayes is an efficient classification algorithm, capable of producing a model within a single pass over the data. Its robustness carries over well for larger data sets as well. Thanks to these features, it is a popular baseline model for text classification.

3.2.3 Linear Support Vector Machines

Another approach to text classification is using Linear Support Vector Machines (SVM). It’s also a linear binary classifier (like Naive Bayes), but (unlike Naive Bayes) it is considered to be non-probabilistic. SVM maps the given examples to points in space. Every data point is considered to be an n-dimensional vector. What the SVM does, is that it searches for a hyperplane (or a set of thereof) that separates the vectors with the greatest possible margins.

SVMs can be used for classification is the labels are available (an example of supervised learning) or for clustering, should the labels be missing (unsupervised learning) [3].

Support Vector Machines are a considerably more complex method than Naive Bayes, usually yielding better results when operating in higher dimensional spaces.

3.2.4 Multi-class classification

Binary classification techniques are effective when dealing with classification into specifically two classes, but that is not always the case. As soon the number of classes grows, reaching three or more, it is referred to as a multi-class classification problem and the need for new approaches arises.

3.2.5 Multi-label classification

Multi-label classification problems are a special subset of multi-class classification. Number of classes is still bound to be greater than two, but as opposed to multi-class classification, where the document ends up being classified into exactly one class, multi-label classification problems have no inherent limit as to how many classes might the document belong to.

The problem of TED talk topic identification is a good example of multi-label classification, as there are multiple classes (topics) in general, but also any document can belong to any number of them — i.e. it is not uncommon to see a talk dealing with a broader range of topics, e.g. “healthcare”, “technology” and “psychology” at the same time.

One-vs-rest

One-vs-rest classifier makes it possible to train a multi-class (or a multi-label) classifier by fitting one classifier per class (as opposed to just one classifier for the entire data set). For each of these classifiers, the currently considered class is fitted against all the others as if they all belonged to one class.

The data set used for the preparation of topic identification model contains multiple tags per document (i.e. talk), therefore, when building the classifier, the topic identification problem must be considered to be a multi-label (which is also inherently a multi-class) one.

To address this, the classifiers were built using the one-vs-rest mechanism. More details about the database of document metadata are given in Chapter 4.

Chapter 4

Implementation details

4.1 TED-LIUM corpus

For the purpose of training the speech recognition model, the TED-LIUM corpus (version 2) was used. This corpus consists of

- 1495 audio talks in NIST sphere format (SPH)
  - Overall, this translates to 207 hours of audio data — 141h of male and 66h of female speech
  - Mean duration a talk is 10m 12s
  - 1242 unique speakers
  - 2.6 million words
  - SPH format info
    - Channels: 1
    - Sample rate: 16kHz
    - Precision: 16-bit
    - Bit rate: 256k
    - Sample encoding: 16-bit Signed Integer PCM
- 1495 transcripts in STM format
- Dictionary with pronunciation (159848 entries)
- Selected monolingual data for language modeling from WMT12 publicly available corpora

Information in this section comes from [21]. At the time of writing there was a newer version available (version 3) with higher count of audio talks and transcripts (2351 in the third version compared to 1495 in the second one). The new data only available in the third version of the corpus were used as a test data set, to generate and classify the transcripts.

The original TED-LIUM corpus has been developed by the LIUM in 2011 (Laboratoire d'Informatique de l'Université du Mans) and was composed of 118 hours of speech and associated transcripts. The corpus was developed by extracting videos and their respective closed captions from the TED website [1]. The provided captions are not verbatim transcripts though, and as such they do not contain disfluencies like hesitations or repetitions.

Another issue with using closed captions directly when building an ASR system is that they are adapted to on-screen reading, and as such are lacking detailed timing information. Fortunately this issue is addressed in TED-LIUM corpus and the timings are retroactively generated from the available data [20].

4.1.1 TED Talks tags

Every TED Talk comes with a set of tags annotating the general topic(s) of the talk. Number of tags for a talk is arbitrary and differs from talk to talk, but every talk has at least a single tag assigned.

4.1.2 Labeling

In order to prepare the classifiers, it is necessary to convert the tags into labels. Because of the arbitrary nature of the tags, there is no standardized representation of the relevant topics, leading to redundancies and overlaps. It is preferable to group related tags under shared labels (e.g. grouping the tags “technology”, “science” and “molecular biology” under a shared label called simply “technology”), since treating each tag as a separate label would inadvertently mean ending up with insufficient data for some of the more obscure tags. At the same time, one tag can belong to multiple labels (e.g. tag “transportation” belonging to “technology”, “environment” and “cities”).

The act of labeling is an essential step in any supervised learning technique, since a source of truth is required in order to establish a baseline upon which the models will be trained. In this case the act of supervision is twofold — the initial list of tags is prepared by a human, and the tags are subsequently aggregated under shared labels by another human (in this case, myself). Since the act of tagging (or labeling) leaves considerable room for interpretation, this, too, can skew the final classification results — no matter how precise the learning model gets, if the label it is trying to predict is incorrectly assigned, the precision is going to be poor. This can be tricky to notice or measure, since this kind of issue would not project itself onto the accuracy scores when comparing the testing and training data. However, if shown to another human (depending on the degree of mismatch), the incorrect topic may be immediately apparent. Because of this, using a reliable corpus is crucial in order to build an effective classification model.

4.2 Data pre-processing

After the tags get aggregated into labels, it is necessary to transform the labels into a representation that the machine learning methods can work with — vectors of binary values instead of strings.

At the same time, it is necessary to extract and reformat the author’s name and the publication year information into a separate attribute that will be used to map the metadata to the contents (transcripts) of the talks. This is a necessary step because of the way the data set containing metadata is formatted.

$$\langle \text{firstName, lastName, MM/DD/YY} \rangle \longrightarrow \text{<firstName>\_<lastName>\_20YY} \tag{4.1}$$

For example, Al Gore’s talk published on 6/27/06 would be formatted as AlGore_2006, making the mapping onto the Kaldi’s transcript files simpler.

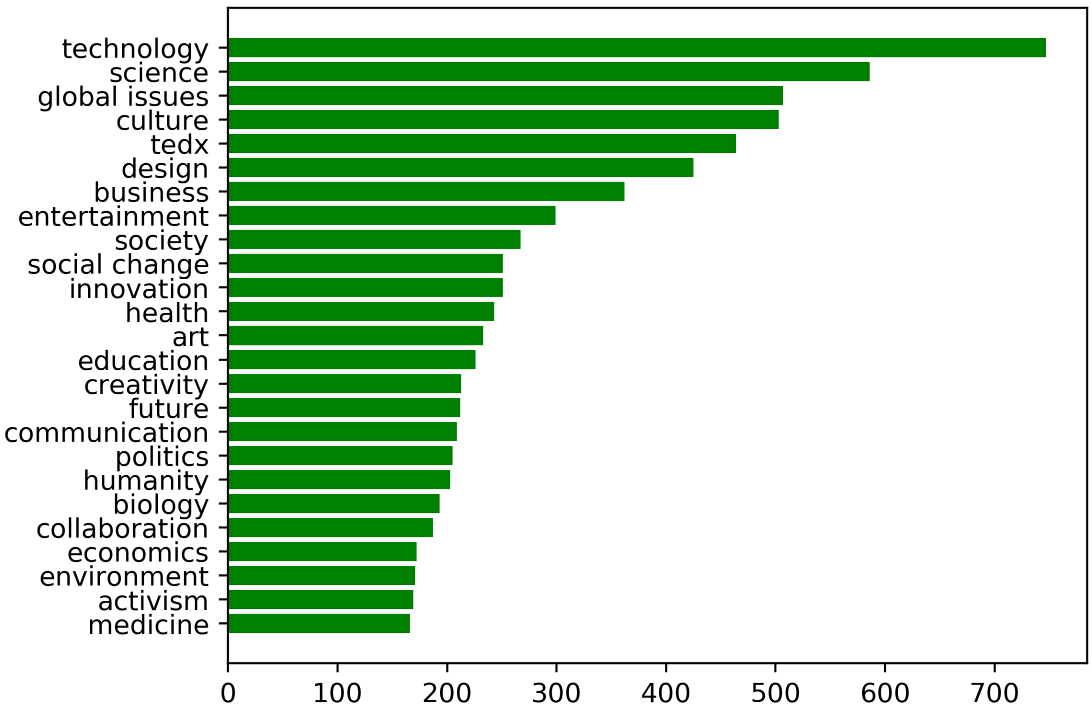

Figure 4.1: Histogram of the 25 most common tags within the used corpus.

4.3 Transcript cleaning

Since the transcripts coming from Kaldi's ASR model contain extra information irrelevant for the purpose of classification, like timestamps, it is preferable to remove it. Here is an example excerpt from the transcription data, provided by the TED-LIUM corpus

BillGates_2009 1 BillGates_2009 16.64 29.31 <o,f0,male> last week talking about the work of the foundation sharing some of the problems and warren buffet had recommended i do that being honest about what was going well what wasn't and making it kind of an annual thing
BillGates_2009 1 BillGates_2009 29.96 43.40 <o,f0,male> a goal i had there was to draw more people in to work on those problems because i think there are some very important problems that don't get worked on naturally that is the market does not drive
BillGates_2009 1 BillGates_2009 43.86 55.06 <o,f0,male> the scientists the communicators the thinkers the governments to do the right things and only by paying attention to these things and having brilliant people who care

Text marked in bold is not essential for classification, so the next step is removing it.

A simple regular expression will be sufficient to handle the necessary substitution (or, in this case, omission). This can be facilitated by Python's `re` library like so;

```
def remove_timestamps(transcript):
    return re.sub('(\n.*\>|^.*\>)', '', transcript)
```

it is necessary to invoke this function for each transcript in order to omit the redundant data.

After building the ASR system, the produced transcripts are following a different format

BillGates_2010-0001586-0001998 i'm going to talk today about energy and climate
BillGates_2010-0001998-0003489 and that might seem a bit surprising because my full time work of the foundation is mostly about vaccines and seeds about the things that we need to invent and deliver to help the poorest two billion live better lives
BillGates_2010-0003489-0004472 but energy in climate are extremely important to these people in fact more important than to anyone else on the planet
BillGates_2010-0004472-0005330 the climate getting worse means that many years their crops on parole will be too much rain not an oft rain

All transcriptions are initially located in a single file, which are then isolated by the speaker, every talk being allocated a single file. This makes it easier to handle and process the transcripts for the subsequent topic identification modeling.

The minutiae of data cleaning and organizing are further described in the attached Jupyter Notebook.

4.4 Model training

For the purpose of this paper, two classifiers will be considered: Multinomial Naive Bayes (3.2.2) and Linear Support Vector Machine (3.2.3).

Models were trained using the implementations provided by scikit-learn [16], using the methods described in chapter 3. Individual steps are described in-detail in the attached Jupyter Notebook.

4.5 Hyperparameter tuning

Before the process of model training is initiated, it is usually necessary to provide a set of parameters for the model — this is applicable for most machine learning models, save some of the simplest ones. The parameters themselves are defined by the model in question, as different models follow different approaches, therefore using different sets of parameters.

There is a fundamental difference between the parameters of the models and the parameters of the training data. Hyperparameters are the parameters of a prior distribution, as opposed to the parameters of the data set that's being modelled. In other words, the hyperparameters are chosen according to a prior expectation (assumption or a heuristic), and define the learning process itself. Model parameters, on the other hand, are determined during the learning process. Choosing the right hyperparameters has a non-trivial impact on the accuracy of the resulting model.

There are many approaches to follow when tuning the hyperparameters, including manual ones — often following a sort of rule-of-thumb logic [11], which tends to be far from optimal. While it may be tempting to skip this part of the model training, as it can be time-consuming and computationally expensive, in favor of relying on the parameter values provided by the library, addressing hyperparameter search properly can lead to drastic improvements in the accuracy of the resulting models.

4.5.1 Cross-validation

Cross-validation is a method for establishing the general fitness of the machine learning model, i.e. seeing how well it generalizes to an independent data set — estimating how well would the trained model perform in practice.

Cross-validation helps to establish whether the right features are being evaluated, as well as whether or not is the model suffering from under or over fitting[1] or is picking too much noise. This is done by partitioning the data set into training and testing data — the model is built using the training partition and then have its accuracy compared against the testing partition of the data set.

K-fold cross-validation

One of the most commonly used methods for cross-validation, the k-fold cross-validation, also referred to as rotation estimation, works by splitting the data set D into k equally sized, mutually exclusive subsets D_1, D_2, ..., D_k [12]. The model is then trained k times, using each subset for validation exactly once. The results are then averaged to produce a more robust estimate of the model's accuracy.

4.5.2 Grid search

Grid search is a method for an exhaustive search (essentially a brute-force approach) over the provided hyperparameter sets, searching for the optimal combination - the one that yields the best accuracy of the resulting model.

Grid search relies on a performance metric, for example the resulting model's accuracy. The exhaustive approach employed by this method means that the search space (essentially a Cartesian product of all hyperparameter combinations) grows rapidly (multi-linearly) as the number of hyperparameter combinations grows, suffering from a phenomenon known as the *curse of dimensionality*. This is offset, to a limited extent, by the fact that the process of training multiple models is usually parallelizable with negligible effort, as the training processes are independent of each other. The problem of high dimensionality is made more severe by the subsequent need for cross-validation, which expands the search space by another dimension. This can quickly lead to having to retrain the same model, with different hyperparameters, thousands of times, to exercise all possible combinations. Depending on the data set and the models being trained, this can become an extremely slow process. There is a number of methods addressing this issue (random search, Bayesian methods), but for the data set used in this paper, using grid search was still a viable option, having to fit 10,368 iterations for the topic identification models — a process that took approximately three hours on a personal computer, powered by a six core processor.

4.6 Results

In order to compare the accuracy and usability of the prepared ASR system for the purpose of topic identification, two data sets were tested. For the first one, transcripts were taken directly from the TED-LIUM corpus (the transcripts that perfectly match the official closed captions released by TED), therefore a close to 100% accuracy can be assumed. The data set was then further split into a training and testing partitions, with 90% of the available data points being allocated to the training partition.

From the results, observable in detail at 4.1, a conclusion can be made that using linear support-vector machines 3.2.3 yields better average accuracy than multinomial Naive Bayes

[1]Both are symptoms of a badly fitting mode - either not fitting the precisely enough (under fitting) or fitting the training data too closely, while lacking the flexibility to generalize over to an independent data set

for all, save three labels. The average topic identification accuracy, between both identifiers, reached 85%.

The second data set has been from the third version of the TED-LIUM corpus [9], the most recent one available at the time of writing. The ASR system and the topic classifiers were trained on the second version of the corpus — which made the new audio files from the third version of the TED-LIUM corpus available as a testing data set, for both the ASR transcription and subsequent topic identification. Cross-matching the generated transcripts with the talk metadata data set (necessary for labelling), provided 760 talks for accuracy testing purposes. Since the models were trained on an isolated group of TED talks, it wasn't necessary to split the testing data set, making its entirety available for the models' testing. The table describing the topic identification accuracy ratios over the ASR-generated data set is shown at 4.2. Accuracy for this data set is affected by the word error rate introduced by the transcription process, which has impact on the topic identification accuracy as well. Substitutions, insertions or deletions, all can skew the topic identification success rate. In this case, however, the averaged accuracy was only 2% lower when compared to using the original transcripts. It can be observed that, since the word error rate of the produced ASR system has been 16.6%, the problem of topic identification can be effectively addressed even with imperfect transcriptions.

Comparing the two topic classification models reveals that the linear SVM model yields a considerably better classification accuracy when compared to the multinomial NB for the majority of labels. While the count of documents available for individual labels differed, an overall trend suggests that using linear support-vector machines yields better results for similar use-cases.

Table 4.1: Prediction accuracy over original TED-LIUM transcripts

Label	Multinomial NB accuracy	Linear SVM accuracy
technology	62.85%	**80.0%**
global_issues	80.0%	**91.42%**
culture	74.28%	**77.14%**
design	**85.71%**	88.57%
business	77.14%	**80.0%**
entertainment	**65.71%**	**65.71%**
society	65.71%	**80.0%**
art	65.71%	**74.28%**
politics	**80.0%**	77.14%
environment	82.85%	**91.42%**
health_care	**74.28%**	**74.28%**
history	**82.85%**	**82.85%**
music	94.28%	**97.14%**
cities	**100.0%**	**100.0%**
war	**97.14%**	**97.14%**
psychology	**85.71%**	88.57%
personal_growth	**94.28%**	**94.28%**
evolution	**97.14%**	**97.14%**
philosophy	**91.42%**	**91.42%**
space	**88.57%**	**88.57%**
math	**94.28%**	**94.28%**

Table 4.2: Prediction accuracy over ASR-generated transcripts

Label	Multinomial NB accuracy	Linear SVM accuracy
technology	63.29%	**69.74%**
global_issues	78.16%	**79.87%**
culture	**75.92%**	72.76%
design	86.18%	**88.03%**
business	80.39%	**87.37%**
entertainment	86.32%	**87.11%**
society	68.42%	**69.34%**
art	73.16%	**80.26%**
politics	71.97%	**80.39%**
environment	67.5%	**78.42%**
health_care	72.63%	**80.66%**
history	**92.89%**	**92.89%**
music	90.26%	**91.32%**
cities	92.24%	**95.92%**
war	92.24%	**93.03%**
psychology	77.11%	**77.76%**
personal_growth	**77.89%**	**77.89%**
evolution	**97.24%**	**97.24%**
philosophy	**88.82%**	88.68%
space	87.24%	**89.61%**
math	**97.5%**	**97.5%**

Chapter 5

Conclusion

The goal of this thesis was to develop two systems for the purpose of natural language processing — one being an ASR system for generating the transcripts of audio streams and the other one a topic classification system for assigning the transcripts produced by the first model into their related labels or categories.

Both the systems were modeled using the TED-LIUM corpus (described in 4.1), which means, mainly for the ASR part, having a system that is well-adapted to "understand" (i.e. transcribe) audio streams with single English speakers (due to the nature of TED Talks). It might be interesting to test the model's accuracy using a different corpus, one with different speakers and environment, but preferably still in English as the language models tend not to carry over to different languages very well.

Another step that could be taken in order to improve the accuracy is using the third version of the TED-LIUM corpus, which is considerably richer than its predecessors (452h of speech data in comparison to 207h in the second version). The ASR modelling process could also be extended by newer machine learning methods, for example replacing the Gaussian mixture models with deep neural networks.

As for the topic identification model, the amount of data available (intersection of the available transcripts with the available metadata corpus) wasn't ideal, a fact that somewhat limited the number of applicable methods. Having a bigger data set of labeled data could make it possible to experiment with deep learning methods for the purpose of classification. It is unlikely to find a big enough corpus directly related to TED data, but the classification model could also be trained on some other corpus containing categorized text data. Another option is to abandon the supervised learning approach to the talk labeling altogether, experimenting with an unsupervised variants instead. Building the labels from scratch, using various topic discovery methods, could lead to removing the reliance on the limited labeled corpora. That could result in better topic identification accuracy, as well as greater flexibility.

The topic identification accuracy could be further improved by tuning the hyperparameters of the current models, as well as experimenting with other machine learning approaches.

Bibliography

[1] TED: Ideas worth spreading. https://www.ted.com. accessed: 2019-05-11.

[2] Ali, A.; Renals, S.: Word error rate estimation for speech recognition: e-WER. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018. pp. 20–24.

[3] Cortes, C.; Vapnik, V.: Support-Vector Networks. *Machine Learning*. vol. 20, no. 3. Sep 1995: pp. 273–297. ISSN 1573-0565. doi:10.1023/A:1022627411411. Retrieved from: https://doi.org/10.1023/A:1022627411411

[4] Dave, N.: Feature extraction methods LPC, PLP and MFCC in speech recognition. *International journal for advance research in engineering and technology*. vol. 1, no. 6. 2013: pp. 1–4.

[5] Davis, K. H.; Biddulph, R.; Balashek, S.: Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*. vol. 24, no. 6. 1952: pp. 637–642.

[6] Guyon, I.; Gunn, S.; Nikravesh, M.; et al.: *Feature extraction: foundations and applications*. vol. 207. Springer. 2008.

[7] Hannemann, M.: Weighted Finite State Transducers in Automatic Speech Recognition. April 2015.

[8] Hazen, T. J.; Richardson, F.; Margolis, A.: Topic identification from audio recordings using word and phone recognition lattices. In *2007 IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*. Dec 2007. pp. 659–664. doi:10.1109/ASRU.2007.4430190.

[9] Hernandez, F.; Nguyen, V.; Ghannay, S.; et al.: TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*. Springer. 2018. pp. 198–208.

[10] Hinton, G.; Deng, L.; Yu, D.; et al.: Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*. vol. 29. 2012.

[11] Hsu, C.-W.; Chang, C.-C.; Lin, C.-J.; et al.: A practical guide to support vector classification. 2003.

[12] Kohavi, R.; et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, vol. 14. Montreal, Canada. 1995. pp. 1137–1145.

[13] Lee, K.-F.: *Automatic speech recognition: the development of the SPHINX system*. vol. 62. Springer Science & Business Media. 1988.

[14] Levenshtein, V. I.: Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, vol. 10. 1966. pp. 707–710.

[15] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press. 2008. ISBN 0521865719, 9780521865715.

[16] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; et al.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. vol. 12. 2011: pp. 2825–2830.

[17] Ponte, J. M.; Croft, W. B.: *A language modeling approach to information retrieval*. PhD. Thesis. University of Massachusetts at Amherst. 1998.

[18] Povey, D.; Ghoshal, A.; Boulianne, G.; et al.: The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. Dec 2011. iEEE Catalog No.: CFP11SRW-USB.

[19] Rabiner, L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*. vol. 77, no. 2. 1989: pp. 257–286.

[20] Rousseau, A.; Deléglise, P.; Esteve, Y.: TED-LIUM: an Automatic Speech Recognition dedicated corpus. In *LREC*. 2012. pp. 125–129.

[21] Rousseau, A.; Deléglise, P.; Estève, Y.: Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. 05 2014.

[22] Salton, G.; Yang, C.-S.: On the specification of term values in automatic indexing. *Journal of documentation*. vol. 29, no. 4. 1973: pp. 351–372.

[23] Shimodaira, H.; Renals, S.: Hidden Markov Models and Gaussian Mixture Models. 01 2017.

[24] Takenobu, T.; Iwayama, M.: Text Categorization Based on Weighted Inverse Document Frequency. 05 1996.

Appendix A

Contents of the CD-ROM

- TED-LIUM transcripts in a stm directory
- ted_transcripts
  - transcript_logs - Directory with Kaldi transcripts containing logging statements
  - transcripts - Directory with cleaned transcripts, isolated by speaker
  - transcripts.txt - cleaned aggregate transcripts
- LICENSE
- list.txt
- README.md
- talks_v2.txt
- talks_v3.txt
- TED_Talk_jan_11_2018.csv
- topic_identification.ipynb
- v3_only_stm.txt

Appendix B

Graphs

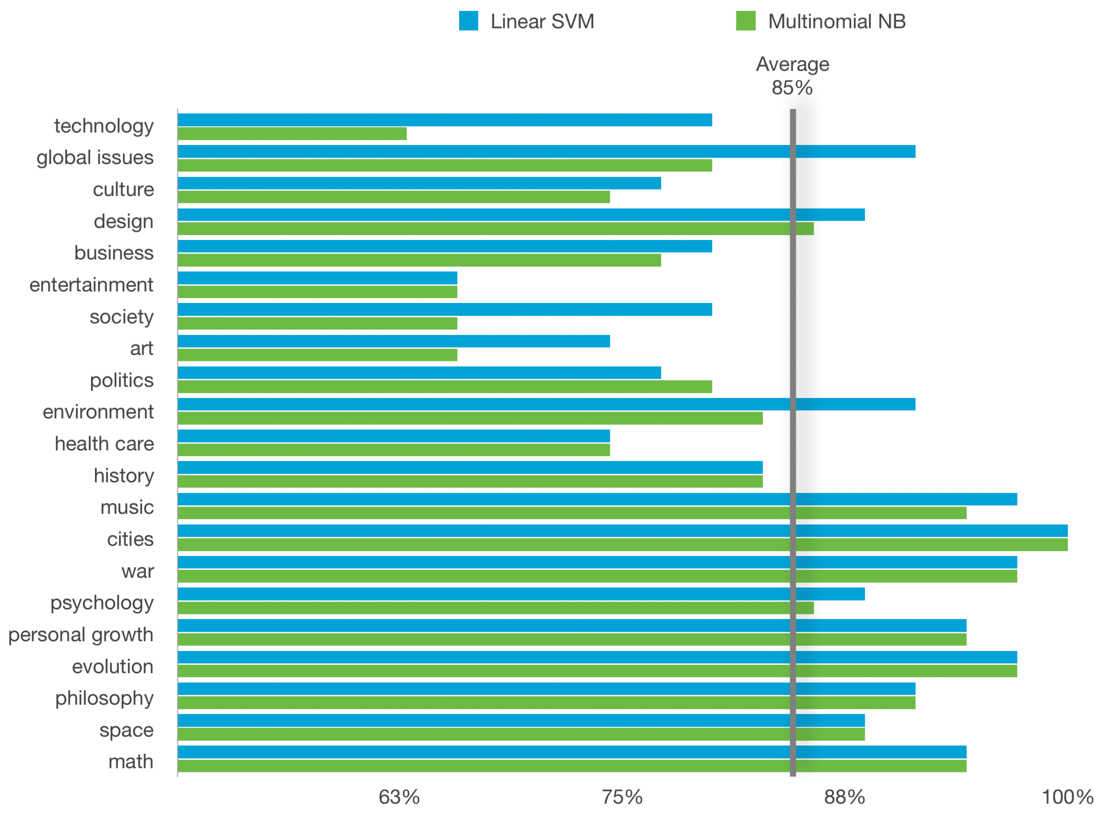

Figure B.1: Model's accuracy using TED-LIUM transcripts

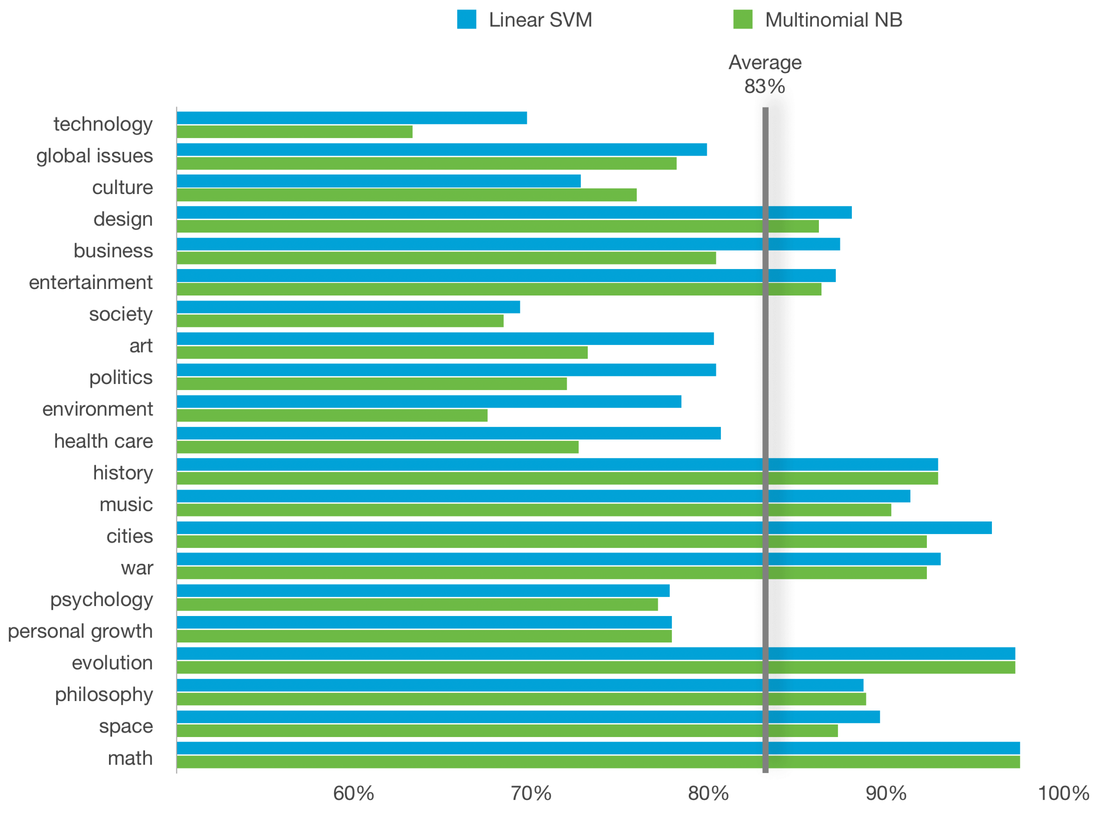

Figure B.2: Model's accuracy using ASR-generated transcripts