NIPS Papers: Topic Modelling and Text Summarization

GAETANO CHIRIACO[1], RICCARDO PORCEDDA[1], GIANMARCO RUSSO[1]

[1] *MSc Data Science, Università degli studi di Milano-Bicocca*

The Neural Information Processing System (NIPS) is a machine learning and computational neuroscience competition held every year from 1987 to date. We tackled a dataset containing all the papers submitted from 1987 to 2017, with the aim of applying unsupervised NLP models for topic modelling (LDA/PLSA) and a supervised approach for extractive text summarization (topic representation and indicator representation).

CONTENTS

1. INTRODUCTION

In an ever data growing era, being able to find and consume data is becoming a real challenge. This report is focused on the discuss and application of two of the most used Text Mining techniques to analyze and extract value from documents: Topic Modelling and Text Summarization. These two methods are applied to a collection of papers for the Neural Information Processing Systems conference (NIPS or NeurIPS).
NeurIPS is one of the top machine learning conferences in the world, it covers topics ranging from deep learning and computer vision to cognitive science and reinforcement learning.
Reading all the papers and understanding their content is a time-consuming and really hard activity for a human being, but by grouping the documents based on their content, using Topic Modelling, and extracting their gist with an unbiased approach, using Text Summarizazion, even such an hard task can became sustainable.
In Section 2 the dataset will be briefly presented and the first needed preprocessing operations will be applied, in order to prepare the data for the next models and analysis.

In Section 3 the first task, Topic Modelling, will be presented. Two models to obtain a topic clustering will be used and compared: LDA and pLSA.

In Section 4 we present a 1D U-Net Architecture for Text Summarization, showing the results on NeurIPS papers and comparing the results with the TextRank algorithm.

2. PREPROCESSING

The corpus used is collection of all 7284 papers presented at the NIPS conference from from the first 1987 conference to the 2017 conference. The initial dataset looked like this:

ID	YEAR	TITLE	EVENT-TYPE	PDF-NAME	ABSTRACT	PAPER-TEXT
1	1987	Self-Organization...	0	1-self-...pdf	Abstract Missing	767 SELF...
10	1987	A Mean ...	0	10-a-mean-...pdf	Abstract Missing	683 A MEAN ...
100	1988	Storing Covariance...	0	00-storing-...pdf	Abstract Missing	394 STORING...
...	...	...	...	...	...	...

The column that were actually used in this study are: "title", "paper-text" and "abstract". So the others have been eliminated. The raw text, stored in the "paper-text" column, needed to pass through some preprocessing steps before being adequately used. The pre-processing pipeline is composed by the following operations:

- Removal of punctuation and digits;
- Convertion of all upper-case letters to lower-case;
- Spaces and character repetition removal;

- Tokenization and Lemmatization;
- Short words removal (less than 3 letters);
- Bigrams and trigrams identification.

Before these steps, another issue must be solved in the raw text. The "paper-text" column includes everything that is readable in the document. For Topic Modelling, the main text (without title, abstract and references) is the only part needed, while for Text Summarization the abstract has to be extracted from the raw text in order to be used as a reference summary. In fact, most of the dataset contains the whole paper texts, including the abstracts, without reporting them in the appropriate "abstract" column.

A. Abstract and Main Text extraction

The main idea behind the extraction process is to separate the paper texts using three keywords: *abstract*, *introduction* and *references*.
Unfortunately, not all papers have the desired sections named with those keywords. In Figure 1 its shown the relative position in the texts of the first occurences of *abstract* and *introduction* and the last occurences of *references*.
It can be seen that the distributions are unimodal, but are also leptokurtic: this confirms the hypothesis that in some cases the desired sections are not named with the chosen keywords, but, still, can be found in other parts of the documents.
To avoid the splitting of the paper text on these false indicators, only the the keywords *abstract* and *introduction* found before the respective 0.95-quantile and the keyword *references* found after the 0.05-quantile are used to split the whole text.
For all the other cases, the documents are removed from the corpus, leaving 6194 documents.

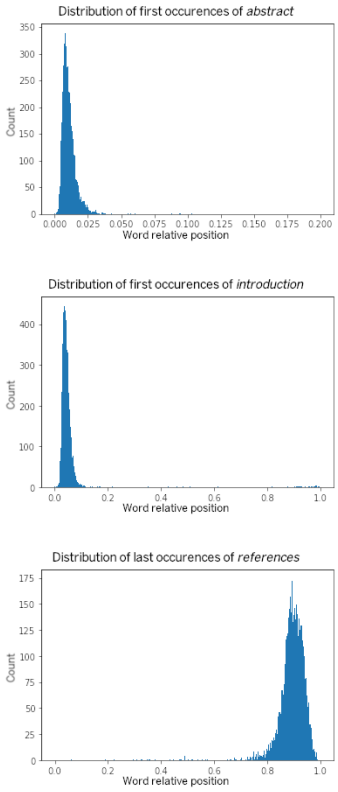

Fig. 1. Distribution of keywords occurrences

B. Sentence level pre-processing and feature engineering for Text Summarization

After the extraction of abstracts and main texts, we subset them into sentences using the NLTK sentence tokenizer and we apply the above mentioned preprocessing steps.
Then, we engineer sentence level features necessary to train our Text Summarizer described in section 4, in particular:

- length of a sentence (number of terms);
- relative position of the sentence in the document;
- distance/similarity measure between document topics and inferred sentence topics (better described in section 3E);
- average of **TF-IDF** weights of every word in the sentence.

Further cleaning of the corpus is made by removing sentences with less than 3 terms. In addition to that, we remove sentences from the main texts with a length greater than the maximum length of sentences in abstracts.

C. ROUGE

Since our aim is the construction of a Text Summarization model, we need an evaluation metric: unfortunately we do not know an ideal reference summaries to compare our own with, but the best choice would be to use as a reference the paper abstracts.
We can do this comparison through **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**.
Here we use ROUGE-2, where a set of bigrams is obtained from the reference summary and from the one produced by the model: if we denote p as the number of bigrams shared by the two texts and q as the number of bigrams extracted from the reference summary, ROUGE-2$= p/q$.

Our fine-grained subsetting allows us to compute ROUGE-2 for each sentence of the main texts by comparing them with every sentence of the paper abstract and choosing the best score.
The relative position of the sentence in the abstract which gained the best score is another indicator that we will later use.

3. TOPIC MODELLING

Topic modelling is an unsupervised machine learning technique that aims at automatically extract word groups that best characterize a set of documents. It provides topics, which are interpreted as a set of words that makes sense together.

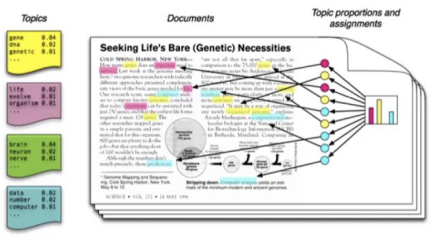

Fig. 2. Topic modelling

TM can be easily confused with text clustering, but the difference is that in text clustering documents are grouped together,

whereas in TM words are grouped together. The main algorithms are Latent Semantics Analysis(LSA) and Latent Dirichelet Allocation(LDA). Considering that LDA seems to be the better performing algorithm (it is a bayesian version of the pLSA), that's what we started with.

A. LDA

When LDA [1] models a new document, it works in the following way:

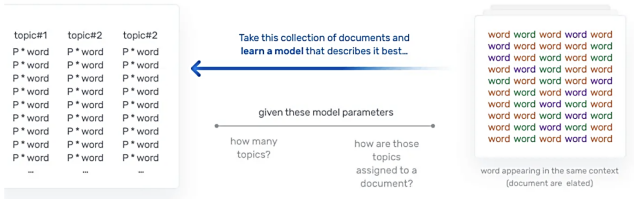

Fig. 3. LDA

LDA is based off of a dirichelet distribution $Dir(p)$, a probability function. The p parameter is named 'concentration parameter' and rules the trend of the distribution:

- Uniform if: $p = 1$
- Concentrated if: $p > 1$
- Sparse if: $p < 1$

When we consider document and topics we consider $p = \alpha$, and for topics and words we consider $p = \beta$. By using concentration parameter $\alpha, \beta < 1$ these probabilities will be closer to what we want to achieve. α and β are respectively document-topic density and topic-word density.

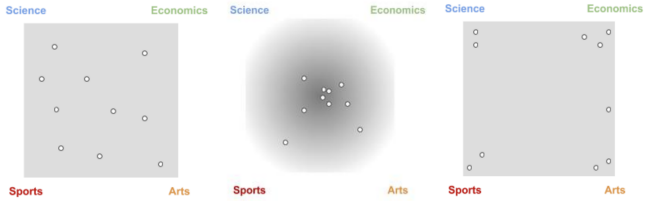

Fig. 4. distribution with respectively: $\alpha = 1, > 1, < 1$

The number of topics is another critical value, usually chosen using the available domain-knowledge or using the **Kullback-Leibler score**.
Given two probability distributions P and Q on the same sample space $\mathcal{X}$, the Kullback-Leibler divergence $D_{KL}(P||Q)$ is defined as

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) log\left(\frac{P(x)}{Q(x)}\right) \quad \textbf{(1)}$$

The LDA model we propose has been initialized with the following parameters:

1. *number or topics*: from 5 to 12,
2. *chunksize*: 10000,
3. *passes*: 15,
4. *iterations*: 100-500-1000,
5. *alpha*: auto,
6. *eta*: auto.

The number of topics has been selected by iteratively trying models with different topics number and comparing the coherence and perplexity scores:

Topics	Perplexity	C_{umass}	C_{uci}	C_{npmi}
5	-8.771	-1.415	0.143	0.021
6	-8.751	-1.391	0.160	0.023
7	-8.732	-1.407	0.207	0.027
8	-8.707	-1.421	0.218	0.029
9	-8.684	-1.469	0.251	0.034
10	-8.675	-1.472	0.423	0.046
11	-8.670	-1.541	0.358	0.041
12	-8.651	-1.537	0.344	0.042

Table 1. Perplexity and Coherence scores

In addition to being the best in 2 out of 4 metrics, the model estimated with 500 iterations and 10 topics seemed to be the one providing the best results.

Fig. 5. LDA topics wordclouds

Lets try to discuss and identify the topics in the wordcloud to see if they make sense:

- *Topic 0*: **Classification**:classifier, label, classification, training set, SVM.
- *Topic 1*: **Scientific Applications**: unit, pattern, dynamic, stimulus, target.
- *Topic 2*: **Clustering**: cluster, clustering, label, user, query.
- *Topic 3*: **Neuroscience**: neuron, signal, stimulus, response, spike.
- *Topic 4*: **Image/Graph**: image, node, graph, object, tree.
- *Topic 5*: **NLP**: prior, topic, document, likelihood, posterior.
- *Topic 6*: **Reinforcement Learning**: policy, agent, action, reward.
- *Topic 7*: **Dimensionality Reduction**: sparse, distance, kernel, norm, subspace.
- *Topic 8*: **Neural Networks**: layer, architecture, image, neural network, gradients.
- *Topic 9*: **Optimization**: loss, estimator, bound, lower, upper.

We are quite satisfied with these results. Anyway, for comparison, we are going to use pLSA too.

B. pLSA

Probabilistic Latent Semantic Analysis [2], the main goal is to distinguish between different context of word usage without using a dictionary or a thesauri. PLSA allow to disambiguate polysemy(words with multiple meanings). It uses a probabilistic method instead of a singular value decomposition(as in LSA). The idea is to find a probabilistic model with latent topics that can generate the data we observe in our document-term matrix. pLSA considers that our data can be expressed in terms of 3 variables:

- **Documents**: $d \in D = d_1, ..., d_N$ observed variables. With N number of documents.
- **Words**: $w \in W = w_1, ..., w_M$ observed variables. With M be the number of distinct words from the corpus.
- **Topics**: $z \in Z = z_1, ..., z_K$ latent(or hidden) variables. Their number K, has to be specified a priori.

$-P(D, W) = \prod_{d,w} P(d, w)$.

Under conditional independence and using the bayesian rule:
$-P(w, d) = \sum_{z \in Z} P(z)P(d|z)P(w|z;)$

$-P(w, d) = P(d) \sum_{z \in Z} P(z|d)P(w|z)$.

The model we're going to comment was initialized with 10 topics and took 138 iterations (Figure 8).
In Figure 9 we can take a look at the topic distribution: the topics seem to be evenly distributed apart from topics 0, 1, 2 and 9.
In Figure 10 we are going to dive into the topics with the help of a wordcloud.

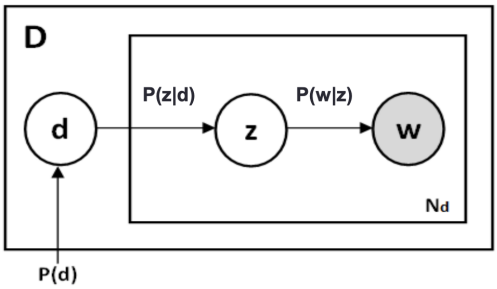

Fig. 6. pLSA

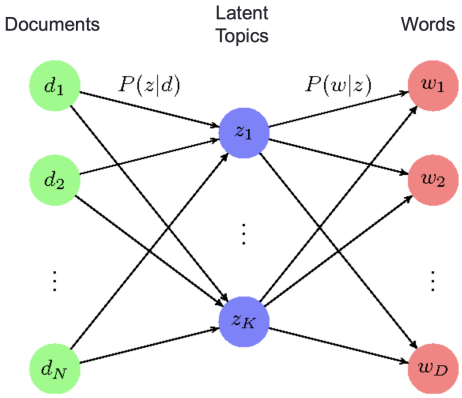

Fig. 7. high level representation of pLSA

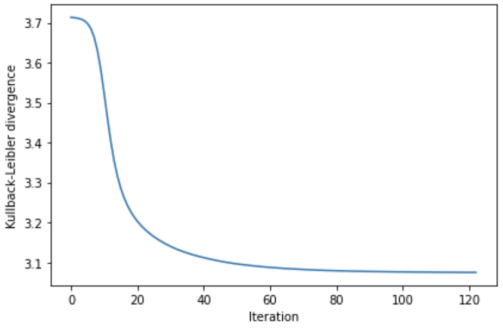

Fig. 8. pLSA's Kullback-Leibler divergence score through iterations

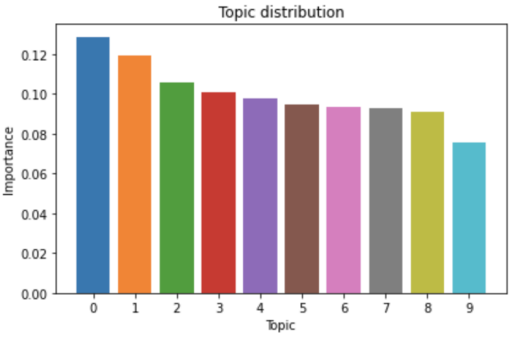

Fig. 9. pLSA topic histogram

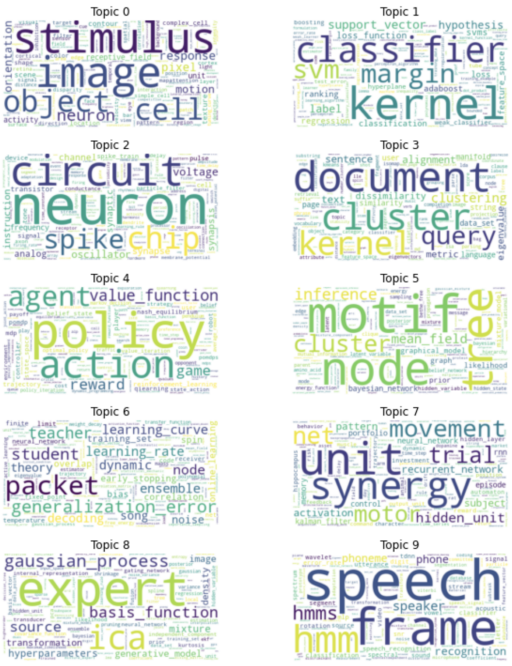

Fig. 10. pLSA topics wordcloud

Some groups are clearly related to well-defined research areas, others seems a bit 'foggy':

- *Topic 0*: **Neurosciences**:neuron, stimulus, motion, cell, response.
- *Topic 1*: **Classification**: classifier, label, svm, ranking, support vector.
- *Topic 2*: **Electrical Engineering**: circuit, voltage, chip, signal, oscillation.
- *Topic 3*: **Information retrieval**: document, query, cluster, text, dissimilarity.
- *Topic 4*: **Reinforcement learning**: policy, action, reward, agent.
- *Topic 5*: **Trees**: tree, node, cluster.
- *Topic 6*: **Unknown**.
- *Topic 7*: **Other Scientific Applications**: synergy, trail, movement, unit, motor.
- *Topic 8*: **Statistics**: gaussian process, ica, bayesian, expert.
- *Topic 9*: **NLP**: speech, hmm, recognition, speaker, phoneme.

We were not able to interpret what topic 6 is about. However, we have to point out that these experiment has been conducted only on a **subset** of the NIPS paper dataset (1000 documents) due to computational limitations (the matrices tend to grow rapidly and memory occupation does too). The lack of performance of our model could be caused or affected by this limitation.

C. LDA vs pLSA

LDA assumes that the distribution of topics in a document and the distributions of words in topics are **Dirichelet distributions**. pLSA (or LSA) does not assume any distribution, and that leads to a less defined vector representation of topics and documents. LDA usually works better than pLSA because it can better generalize to new documents:

- in pLSA, the document probability is a fixed point in the dataset: if we have not seen a document, we do not have that data point;
- in LDA, the dataset serves as training data for the Dirichelet distribution: if we haven't seen that document we can easily sample from the distribution and move forward from there.

LDA is considered state-of-the-art in topic modelling applications[3], and, for what we have seen, we can confirm that: it performs way better than pLSA and with reduced computational costs.

D. Actual NeurIPS Topics

For completeness, we report the topics announced in the NIPS 2017 Call For Papers [4]:

- **Algorithms**: Classification, Clustering, Components Analysis (e.g., CCA, ICA, LDA, PCA), Kernel Methods, Nonlinear Dimensionality Reduction...
- **Probabilistic Methods**: Bayesian Theory, Causal Inference, Distributed Inference, Gaussian Processes, Hierarchical Models, Variational Inference...
- **Optimization**;
- **Applications**: Audio and Speech Processing, Computer Vision, Denoising, Hardware and Systems, Image Segmentation, Information Retrieval, Motor Control, Natural Language Processing, Object Detection, Object Recognition...
- **Reinforcement Learning and Planning**;
- **Theory**: Competitive Analysis, Computational Complexity, Control Theory, Frequentist Statistics, Game Theory...
- **Neuroscience and Cognitive Science**;
- **Deep Learning**;
- **Data, Competitions, Implementations, and Software**.

E. Topic distance

Once the LDA model is trained, the *gensim* library allows for any corpus (even at sentence level) to obtain its topics probability distribution, that in our case is a vector of 10 real values (one for each topic).
We therefore compute this vector for every sentence and one for each document. But, for the summarization task, we would like to have a single number to express the similarity between the topics identified in a sentence and the topics of the document to which the same sentence belongs. We achieve this by computing the **Jensen-Shannon distance**.
It is a metric obtained as the square root of the **Jensen-Shannon divergence**, a symmetrized and smoothed version of the Kullback–Leibler divergence:

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \quad \textbf{(2)}$$

where $M = \frac{1}{2}(P+Q)$.

4. TEXT SUMMARIZATION

The goal is to produce a shortened version of the document, with the obvious advantages of reduced reading times, but also to optimize the search operations and to improve the effectiveness of indexing. Text summarization tasks can produce two types of summarization:

- **Extractive** summarization: important phrases are selected from the input text and used to create the summary;
- **Abstractive** summarization: the model forms its own sentences and use them to create the summary.

The work described in this article only refers to extractive summarization tasks.

Furthermore, based on purpose, we can classify text summarizers as:

- **Generic**: the model makes no assumptions about the domain or content of the text to be summarized and treats all inputs as homogeneous;
- **Domain-specific**: the model uses domain-specific knowledge to form a more accurate summary;
- **Query based**: the summary only contains information which answers natural language questions about the input text.

Since we are dealing with papers of the NIPS conference and we are using topic representation approaches, we are building a domain-specific summarizer.

A. Approaches

Topic representation approaches: first derive an intermediate representation of the text that captures topics discussed in input, then, based on this representation, sentence in the input receive an importance score. The most used algorithms are Latent Semantic Analysis (LSA) and Latent Dirichelet Allocation(LDA). In this case the score is usually related to how well a sentence expresses some of the most important topics in the document.
Indicator representation approaches: the text is represented by a diverse set of possible indicators of performance not aimed at discovering topicality. These indicators are combined, usually using machine learning techniques, to score the importance of every sentence.
Often used indicators: sentence length, location in the document, presence of certain words.

In this work we mix the two approaches and, furthermore, we have been inspired by the research of Zhang, Joo Er and Pratama [5]: the power of Convolutional Neural Networks is getting noticed not only in the Computer Vision field, but also in others research areas like, indeed, Natural Language Processing.

B. Document Signal Representation and Semantic Segmentation

All the tasks completed above allow us to describe all documents with 1D vector representations where each single element of a vector is a sentence indicator.
In fact, what we obtain is a multi-channel signal like the one shown in figure 11. Each sentence is represented by a point in a R^4 space, where the four dimensions are:

- Sentence length: Number of words in the sentence;
- Sentence relative position: Relative position of the sentence in the paper. It's equal to 0 if it's the first sentence in the text and it's equal to 1 if it's the last;
- TF-IDF weight: average TF-IDF value of the words in the sentence;
- Topic distance: calculated using the Jensen-Shannon distance.

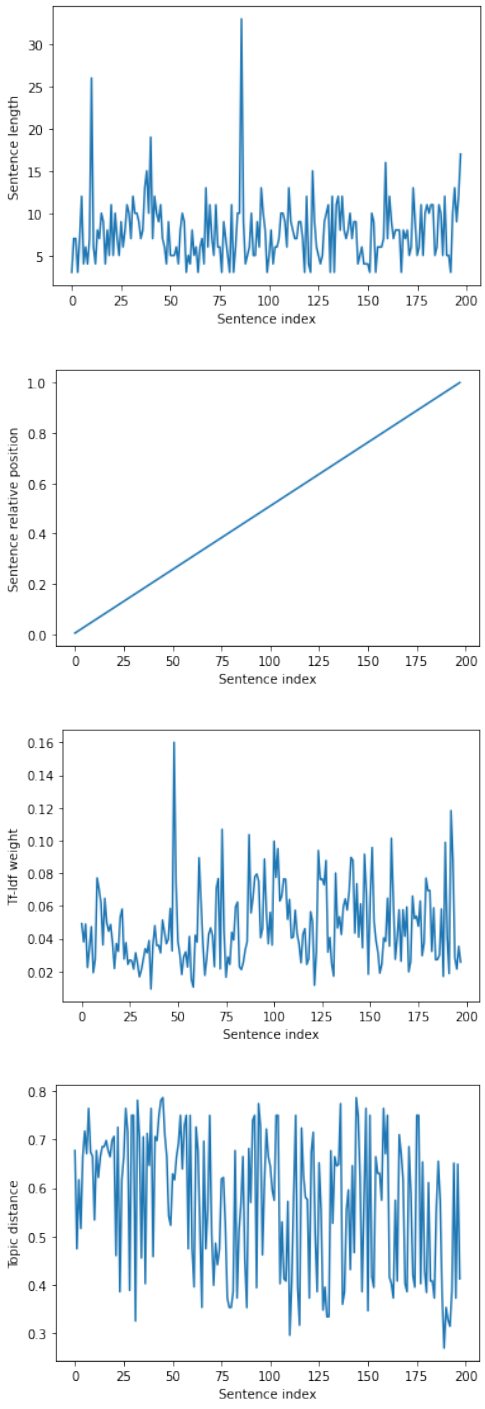

Fig. 11. Multi-channel signal representation of a document

Then all these signals are padded so that every document has the same vector representation length.
Considering the maximum length of the documents in the corpus, we choose a length of $896 = 2^7 \cdot 7$, which is convenient for the model architecture we are going to build.

What we want to obtain is a semantic segmentation of this multi-channel 1D signal that can distinguish between summarizing sentences and non-summarizing ones.
To do that, we exploit the power of Convolutional Neural Networks by building a 1D U-Net architecture.

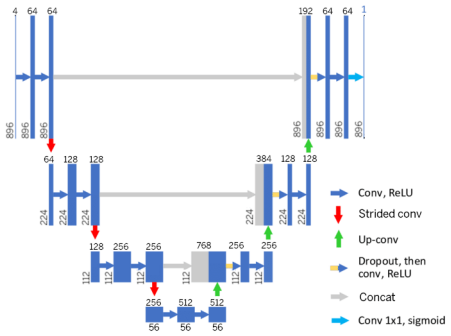

Fig. 12. The 1D U-Net architecture built to identify summarizing sentences

To train this model we use binarized ROUGE-2 target values (1 if ROUGE-2 $\neq 0$ and 0 otherwise) and we define a *Jaccard distance* as our loss function: being y the target vector and $\hat{y}$ the prediction vector, we have

$$J(y, \hat{y}; \rho) = \frac{|y \cap \hat{y}| + \rho}{|y| + |\hat{y}| - |y \cap \hat{y}| + \rho} \tag{3}$$

$$\mathcal{L}(y, \hat{y}; \rho) = \rho[1 - J(y, \hat{y}; \rho)] \tag{4}$$

where ρ is a smoothing factor included to avoid exploding or vanishing gradients.
In this work, we set $\rho = 100$.

Once this model is trained and we have predicted which sentences are the summarizing ones, we use the prediction vector as a new feature for a second model which predicts the best relative position of the same sentences in the summary.
The model architecture is still a U-Net, but this time the loss function is a simple mean squared error (since we must predict real values between 0 and 1).

To sum up what our model does, it predicts which sentences are the summarizing ones, it sorts them and then keeps the first n (in the following examples, $n = 5$).

5. RESULTS

Our model obtains an average ROUGE-2 score of 0.09 with a standard deviation of 0.07. In Figure 13 the distribution of these scores.

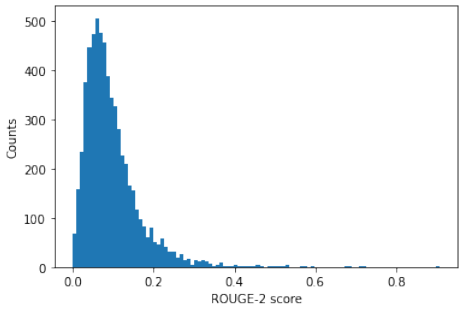

Fig. 13. ROUGE-2 scores distribution of summaries generated by our model

Abstract
The paper suggests a statistical framework for the parameter estimation problem associated with unsupervised learning in a neural network, leading to an exploratory projection pursuit network that performs feature extraction, or dimensionality reduction.

Summary
The search for a possible presence of some unspecified structure in a high dimensional space can be difficult due to the curse of dimensionality problem, namely the inherent sparsity of high dimensional spaces.
Recently, exploratory projection pursuit (PP) has been considered (Jones, 1983) as a potential method for overcoming the curse of dimensionality problem (Huber, 1985), and new algorithms were suggested by Friedman (1987), and by Hall (1988, 1989).
Feature detecting functions of neurons have been studied in the past two decades (von der Malsburg, 1973, Nass et al., 1973, Cooper et aI., 1979, Takeuchi and Amari, 1979).
This paper suggests a statistical framework for the parameter estimation problem associated with unsupervised learning in a neural network, leading to an exploratory PP network that performs feature extraction, or dimensionality reduction, of the training data set.
Neural networks seem promising for feature extraction, or dimensionality reduction, mainly because of their powerful parallel computation.

Here we show the comparison between the best summary produced by our model and the original abstract, with a ROUGE-2 score of 0.91.
The paper is *A Neural Network for Feature Extraction* by Nathan Intrator.

Then, we show one of our worst results, *Sequential Tracking in Pricing Financial Options using Model Based and Neural Network Approaches* by Mahesan Niranjan, with a ROUGE-2 score of 0:

Abstract
This paper shows how the prices of option contracts traded in financial markets can be tracked sequentially by means of the Extended Kalman Filter algorithm.
I consider call and put option pairs with identical strike price and time of maturity as a two output nonlinear system.
The Black-Scholes approach popular in Finance literature and the Radial Basis Functions neural network are used in modelling the nonlinear system generating these observations.I show how both these systems may be identified recursively using the EKF algorithm.
I present results of simulations on some FTSE 100 Index options data and discuss the implications of viewing the pricing problem in this sequential manner.

Summary
A number of problems, including forecasting a univariate time series from past observations, rating credit risk, optimal selection of portfolio components and pricing options have been thrown at neural networks recently.
This is because data arrives at the modeller sequentially, and there is the need to build and apply the 961 Sequential Tracking of Financial Option Prices best possible model with available data.
Hutchinson et al use this to define a highly relevant performance measure Sequential Tracking of Financial Option Prices 965 (a) Estimated Call Option Price 0.25 —True 0.2 Estimaie 0.15 0.1 0.05 O – —— L— —- – —- —-L—-L—- – —- 20 40 60 80 100 120 140 160 180 200 220 (b) Estimated Put Option Price 0.1.—.—.—-.—-.—— .—-.—-.—-.—-.—-,—-. .. . 0.06 0.04 0.02 20 40 60 80 100 120 140 160 180 200 220 time
Figure 1: Tracking Black-Scholes Model with EKF; Estimates of Call and Put Prices
suitable to this particular application, namely the tracking error of a delta neutral portfolio.
The value of volatility that gives the correct options price through Black Scholes equation is called option implied volatility, and is usually different for different options.

Here, in red, we highlight what appear to be outlier sentences that could have misled our model during training (and, consequently, during inference).
Checking the actual paper (Figure 14), we see that these outliers are actually:

- Paper title
- Page number
- Labels, titles and caption of figures

We therefore understand that many documents of our corpus have been digitalized with an OCR technique and simple preprocessing can not handle these situations, since most of the terms are legit words.

A. Brief comparison with TextRank

We try now to generate summaries with a well-known model: TextRank [6]. TextRank is inspired by Google's PageRank: documents are represented as a graph, where sentences form nodes and edges are determined by their similarity (if it is greater than a chosen threshold, sentences are connected).
In particular, TextRank computes the cosine similarity of TF-IDF

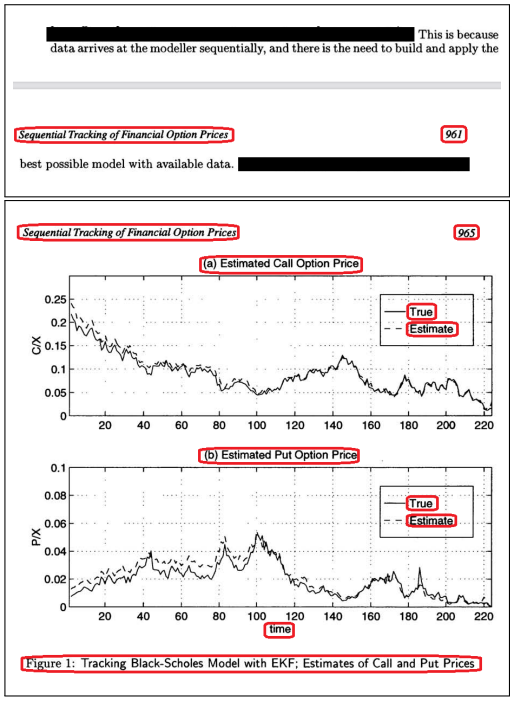

Fig. 14. The actual paper by Mahesan Niranjan

weights.
Graph partitions represent the identified discrete topics and, for each document, the more a sentence is connected, the more important it is and it will be chosen for the summary.

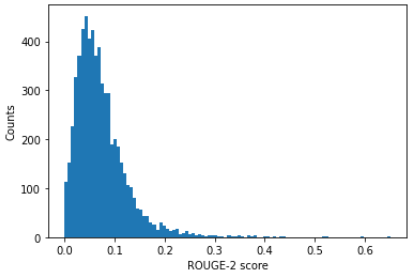

Fig. 15. ROUGE-2 scores distribution of summaries generated by TextRank

The summaries generated by TextRank have an average ROUGE-2 score of 0.07, with a standard deviation of 0.05.
Our model has, then, statistically comparable results.
Moreover, it must be said that TextRank needed over 1.25 hours to produce the summaries for every document in the corpus (results obtained with a Google Colab machine with 2.2GHz AMD Processor), while our model achieved the results shown above in half the time, where most of it (roughly 28 minutes) was used to compute sentence-level ROUGE-2 target scores and TF-IDF weights.

6. CONCLUSIONS

Regarding our topic modelling approach we can confirm what we anticipated in the corresponding section: the LDA model performs better than pLSA and it is also lighter in terms of execution times and computational costs. The topics we got out

of the LDA clearly make sense and are representative of what the articles submitted for NeurIPS are about. On the text summarization side, what we have produced is a single document, domain specific, extractive summarizer. We outperformed a well-known indicator representation approach algorithm that is TextRank.
Certainly we have space for improvement: as shown in the example of the worst summary produced, we could implement more comprehensive preprocessing techniques, include a further semantic sentence-level representation with word embedding models or try to resize document signals (with a well-chosen interpolation) instead of padding them.

REFERENCES

1. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," J. Mach. Learn. Res. **3**, 993–1022 (2003).
2. T. Hofmann, "Probabilistic latent semantic analysis," Uncertainity Artif. Intell. UAI'99, Stock. (1999).
3. C. Z. Yue Lu, Qiaozhu Mei, "Investigating task performance of probabilistic topic models: an empirical study of plsa and lda," (2011).
4. "Nips call for papers," in *https://nips.cc/Conferences/2017/CallForPapers*, (2017).
5. Y. Zhang, J. E. Meng, and M. Pratama, "Extractive document summarization based on convolutional neural networks," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, (2016), pp. 918–922.
6. R. Mihalcea and P. Tarau, "Textrank: Bringing order into text." (2004).