Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Deep neural models to represent news events

Master's Thesis in Computer Science
by

Denys Chechelnytskyy

Internal Supervisor

Assoc. Prof. Vinay Jayarama Setty

Internal Reviewer

Assoc. Prof. Vinay Jayarama Setty

External Reviewer

Dr. Avishek Anand

June 15, 2018

"No problem can be solved from the same level of consciousness that created it."

Albert Einstein

Abstract

The thesis is dedicated to the background linking tasks for news articles, utilizing the deep neural network models. The goal is to retrieve similar articles based on the news story currently viewed. We examined neural and non-neural representations for raw text and discussed notions of similarity a good model should identify and retrieve. We covered various deep neural network models and highlighted their advantages and disadvantages.

Inspired by deep neural architectures in the area of Information Retrieval we adjusted the Deep Semantic Similarity model to the background linking task. Our refactored DSSM architecture employs a convolutional neural network with multiple filters and regularization techniques. This convolutional network acts as an auto-encoder and learns the compressed representations of news articles and news stories. Cosine similarity is used as the proximity metric to retrieve related news articles. Experimental results prove that our adjusted DSSM model is applicable for the background linking task, and overperforms the baseline SVM model.

We discovered that corpora distributions affect the performance of the model. A model trained on news corpus containing mostly political and social news will perform poorly on news corpus about sport and entertainment news. Grid search and hyperparameter tuning are also important. Deep neural network architectures are powerful tools which can be used to solve complicated tasks and approximate nearly any function. Having a good quality dataset is half of the success. The DSSM model is planned to be adjusted to various news corpora and applied to different tasks; such as automatic linking of news articles to Wikipedia pages and linking news articles to news events. We assume this model can be extended to learn representations of a sequence of events for the task of linking background events.

Acknowledgements

First, I will express my profound gratitude to my thesis supervisor, Associate Professor Vinay Setty at the University of Stavanger for dedicated guidance, useful comments, recommended readings, regular meetings, motivation, and patience. I am thankful to Vinay Setty for introducing me to his research topic and guiding me throughout my work. He would always give an advice and help. I could stop by his office to discuss my research, get directions and expert opinion.

I am deeply grateful to Dr. Avishek Anand for expressing interest in the research topic and agreeing to review my thesis.

I will also thank the University of Stavanger for allocated computational resources: servers with GPUs and study facilities and Origin/Corporator AS for allocated office resources for conducting my academic research.

I will thank my friends Diana, Thomas and Tomasz who supported me throughout the entire process and gave me advice. They were there for me when I needed their help and support.

I am particularly grateful to Reidar Huseby for "finding my talent", bright ideas and valuable advice.

Contents

Abbreviations

ACC	**Acc**uracy
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
BM25	**B**est **M**atching 25
CM	**C**onfusion **M**atrix
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
DNN	**D**eep **N**eural **N**etwork
DSL	**D**omain **S**pecific **L**anguage
DSSM	**D**eep **S**emantic **S**imilarity **M**odel
FN	**False Negatives**
FP	**False Positives**
GD	**G**radient **D**escent
GloVe	**Glo**bal **Ve**ctors
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HTML	**H**yper**t**ext **M**arkup **L**anguage
IR	**I**nformation **R**etrieval
JSON	**J**ava**S**cript **O**bject **N**otation
L2R	**L**earning to **R**ank
LM	**L**anguage **M**odeling
LSA	**L**atent **S**emantic **A**nalysis
LSTM	**L**ong-**S**hort **T**erm **M**emory
NN	**N**eural **N**etwork
RBF	**R**adial **B**asis **F**unction

ReLU	**Re**ctified **L**inear **U**nit
REST	**Re**presentational **S**tate **T**ransfer
RNN	**R**ecurrent **N**eural **N**etwork
SGD	**S**tochastic **G**radient **D**escent
SPC	**Spe**cificity
SVM	**S**tate **V**ector **M**achine
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
TN	**T**rue **Negatives**
TNR	**T**rue **N**egative **R**ate
TP	**T**rue **P**ositives
TPR	**T**rue **P**ositive **R**ate
URL	**U**niform **R**esource **L**ocator

Chapter 1

Introduction

1.1 Motivation

Every day hundreds of thousands of news articles from various categories such as politics, economy, sports, natural disasters, science and technology are consumed by hundreds of millions of internet users across the world. In the past decade news is being increasingly consumed online and consequently, print media is on the decline. We were interested to see if we could streamline the process of receiving news by speeding it up for consumers, which would provide agencies a competitive edge and therefore be marketable.

News sources are accessible at all hours of the day and readers have access to different sources of information including the ever popular worldwide sources, national, as well as local or municipal. Readers have the opportunity to take a deeper look at the event, about which they are reading, by accessing news sources from different countries. Usually, different news agencies present the same news events with different takes on it. Some events keep happening in the country of their origin, but vanish from international news feeds because this event is no longer considered newsworthy or significant by the international community.

Readers are able to retrieve full content of the news by analyzing different news articles from different sources. Nowadays historical facts can be verified by conducting research via online media. As such, one can now subscribe to a specific news agency or news service, select categories or events of interest about which to be informed, and get a notification on a mobile device or get emails about new articles as soon as they are being released. Since news agencies and news providers are competing with each other in getting more subscribers they should propose something unique.

We also observed that document classification is no longer a task that should be performed manually by people. The amount of information available is huge, hence, manual classification and categorization can take a significant amount of time, which is basically counter-intuitive to the whole concept of having current and relevant information, not to mention costly.

Document classification is, however, necessary to divide content by theme, category, and relation to certain news events as well as to propose readers relevant reading topics. Classifying news articles creates a common semantic space for conducting research and journalist investigations.

Deep neural networks are gaining popularity due to their ability to perform different tasks with a human level of accuracy. Indeed, sometimes they show even higher accuracy. Deep neural networks learn on their own without the need to select features manually. By using deep learning in the online news industry we can make categorization, classification and news event identification seamless through automation. We believe we can create a marketable tool for industry leaders to use to better serve their clients by enhancing deep neural network capabilities.

1.2 Problem Definition

The online media industry is highly competitive. News agencies constantly look for new ways to attract readers and keep current customers satisfied. Good journalistic skills are only part of producing new online media. In order to keep readers interested, news agencies should propose something different like new services, good prices and efficient searching. Proposing related background articles or recommended reading is a good and very effective trick to satisfy readers' need for news and propose more news to read within the user's area of interest. The goal of retrieving related background articles is to provide similar articles, though not necessarily current, that correspond to the topic of the paragraph or entire article being read. Based on the story, similar news articles will be recommended.

We believe automated evaluation is more beneficial than manual. While accurate, manual evaluation is cumbersome and time-consuming. Something that catches the eye of evaluator can be irrelevant to reader's taste. Process automation is beneficial because it removes biases and saves time.

Readers often see links in between the paragraphs of news article or sections after it with text "Related training", "You may also be interested in the following related posts", "Related articles", etc. Some popular news agencies already have recommended reading

sections. For example, The Washington Post has "The Post Recommends" and "Read more" sections, and BBC has "More on this story" and "Related Topics" sections on their web pages. There is, however, always room for improvement.

For instance, not all online news media outlets have such systems. Many of them just have recommended reading from the category, or 'top stories' retrieved based on user's history. In other words, manually constructed features are used to generate the list of background articles. For example, SkyNews from the United Kingdom has an article with the title "Hawaii eruption: 'Pele, the goddess of fire and volcanoes, is showing herself' "[1] about the volcanic activity in Hawaii, USA. At the end of the article there is a "More from Hawaii" recommended reading section with a list of the following articles: "'Sulphur and caramelised foliage': Living on Hawaii under the threat of Kilauea", "Hawaii's Kilauea volcano spews toxic glass cloud as lava reaches ocean", "Fresh lava flow could block Hawaii escape route", "Hawaii volcano spews lava and ash", "Kilauea volcano: Hawaiians warned of 'powerful' eruptions within hours" and "Volcanic activity still rocking Hawaii". It is easy to notice that all of these articles come from category "Hawaii". What if the user wants to read about current volcanic activity worldwide or eruptions caused by volcanic activity in other countries? This example shows that not all proposed results are good and precise; errors happen. We all know posting irrelevant or false content can harm a news agency's reputation, but so can not posting relevant and related stories. In practice, it is useful to retrieve related background articles to provide readers with interesting reading. As in the example above, the scope of related articles has many more possibilities.

Another issue we face is that news articles as text documents have a hierarchical structure. Each word is composed of characters, sentences are composed of words, articles are composed of sentences. Various words contribute differently. In other words, not all words are as important as others. The same applies to sentences. Not all sentences are of the same importance to a news article's content.

The goal of this work is to build a model with deep neural architecture for retrieving relevant background articles which will recommend more relevant articles by utilizing the hierarchical structure to identify useful semantics and create connections to more material.

[1] https://news.sky.com/story/hawaii-eruption-pele-the-goddess-of-fire-and-volcanoes-is-showing-herself-11382374

1.3 Usecases/Examples

Our model can be used by news agencies to suggest related, interesting news articles to readers. Journalists can retrieve similar articles to produce more meaningful content, displaying and highlighting some information, facts, and evidence that is missing in other articles. This model can be used by researches to research information in the sphere of news. This model can be also used by everyday internet users who want to find similar news articles or research the event in more depth.

Usages of the model:

1. Event classification - whether a given article is a political or technological event.
2. Event similarity - retrieve all similar events. For example, get events which are similar to NBA Championship like other sports tournaments.
3. News article similarity - retrieving related background articles. For example, suggest news about global warming while reading the article about vanishing population of polar bears.

1.4 Challenges

The original idea of neural networks comes from neural science - the science about the human brain. The goal of artificial intelligence and machine learning is to build a program (model) which will show the same or a higher level of accuracy when compared to predictions made or results created by actual human beings.

Think of a child who has not been in school yet. You cannot expect the child to tell you about programming and calculus because the child has never learned anything about it. Same applies to machine learning and artificial intelligence. In order to build a model we first need to teach it on a large training set. The size of the training set matters. The more examples the model learns, the more accurate predictions it can make.

Here comes the first challenge - obtaining a ”good” training dataset. By saying ”good” we mean utilizing news articles that span a long period of time and are from reliable sources and are about different news events. This need to be done without noise from web crawlers. In case of small training dataset, we have a high probability of over-fitting the model. To address the over-fitting problem, regularization techniques should be used such as L1 and L2 regularization, dropout, bagging (ensemble), early stopping, data augmentation, etc.

To get the training dataset we need to use web crawlers: special programs which visit page by page and retrieve useful information. There are some datasets available on the global net retrieved manually by researches. However, those datasets are usually outdated by the time they are released.

We must, therefore, create new datasets by building new parameters in relation to article length and dates of coverage. Some news articles are very short and consist of 4-5 sentences, while others consist of 20+ sentences. At the same time, it is hard to predict how many sentences are really useful. The amount of news articles provided to cover the news event also matters as does the time period of coverage. For example, if we have a dataset with news articles from the same news event from 2009 until 2018, with a gap from 2011-2012 that means that our model is likely to misclassify the news article from 2012 about the same event. As such, we need to build datasets carefully. The second challenge faced is resources and time consumption. Most machine learning algorithms are not new and come from the 1990s [1]. Due to a lack of computational power (resources), these algorithms could not be tested. Even today training the model is a time and resources consuming process. For example, training one model on CPU can take days or even months. Model architecture and parameter tuning require many training and validation runs. Recently it has been discovered that simple mathematical operations can be done efficiently by graphics cards (GPUs) and faster than on CPU. However, the speed of the training process still depends on the memory of GPU and the number of GPUs which are synchronized. By using modern GPUs, the training process can be shortened to minutes and hours. We trained our model on Nvidia Tesla GPUs with 12-16 GB of memory (credits to the University of Stavanger for provided resources). Indeed, GPUs are located on the remote server, which means we need to remotely connect to the server. As a result, there is a delay and some discomfort in maintaining the code on the remote server.

Another problem we are faced with is how to evaluate the results. If the training had been done on the outdated big dataset, we do not believe it would be useful as it would not correlate to new current testing data. The current model is adjusted to news articles in English with a limited number of news events. It is hard to predict how the model will behave on the unseen news events. Some problems can arise when news events are correlated and are the part of the bigger news event.

1.5 Contributions

Deep learning had been used to classify documents, but no approaches are known for news articles. We propose the usage of the Deep Semantic Similarity Model originally

designed for information retirement tasks for the retrieval of related background news articles. The idea is to learn news article vector representation and by means of similarity teach the model to differentiate articles within the same stories.

We make the following contributions:

1. We use Deep Neural Networks to solve news background linking task.
2. We apply Support Vector Machine as a baseline model for solving background linking task.
3. We use different sampling techniques to generate news articles & new event pairs by utilizing Deep Similarity Semantic Model.
4. We present experimental results which demonstrate the benefits of using deep neural architectures over standard machine learning algorithms.

1.6 Outline

The remainder of this thesis has the following structure. Chapter 2 contains a short overview of text categorization approaches in machine learning word. We also point out differences between neural networks and deep neural networks and describe some popular deep neural network architectures. Chapter 3 is dedicated to raw text representations in machine learning, notions of similarity and similarity metrics, as well as traditional and neural approaches in Information Retrieval. In Chapter 4, we introduce the Deep Semantic Similarity Model adjusted for the linking of news articles based on background information. In Chapter 5, we describe the datasets, present experimental results and discuss some issues on the topic. Chapter 6 summarizes the outcome of research and highlights future work directions.

Chapter 2

Background concepts

Before we tackle the problem of background linking task, we should have some basic understanding of machine learning principles. In Section 2.1, we demonstrate why machine learning is important by the document classification example and introduce different types of learning. In Section 2.2, we list some popular feature-based classifiers, and put emphasis on Support Vector Machine, as we will use it as the baseline model. We describe neural network and deep learning paradigm in Section 2.3. We are giving some insight on feedforward, convolutional and recurrent neural networks in Sections 2.4, 2.5 and 2.6 correspondingly.

2.1 Document classification

Document classification is a problem of assigning a label, a class, a category to a text document. The problem of document classification became more popular in the digital era. With the fast growth of digital information, the number of electronic documents stored online increases exponentially. Document classification can be performed manually or automatically (algorithmically).

Manual document classification is done by people, researches, and a qualified staff. There are governmental organizations which manually classify documents for information retrieval, for example, Text Retrieval Conference (TREC) Assessors from the National Institute of Standards and Technology (NIST). Manual document classification is a rather expensive process, especially when done by experts. In the search for cheaper and still reliable ways of manual document classification, crowdsourcing platforms have been introduced. A crowdsourcing platform allows users with different backgrounds and education, to manually classify text documents for a small fee. Example platforms are

Amazon Mechanical Turk in USA and Crowdflower in European Union[1]. Crowdsourcing platforms use large crowds of people to speed up the classification process.

Automatic document classification is performed by computers, and other electronic computational devices based on some algorithms. Digital society nowadays moves form algorithmic classifiers to AI (Artificial Intelligence). AI approaches are very attractive because they scale well and tend to provide a human level of accuracy in classification tasks. Automatic document classification can be divided into supervised, unsupervised, and semi-supervised document classification [1].

Supervised document classification uses labeled training data. The result and classification accuracy mostly depend on the quality and amount of entries in the training dataset and the amount and quality of features. A part of a dataset is preserved for validation and testing purposes. The outcome of the model is predictable, inner relations in data can be predicted and estimated with manual features. For example, if pressure is low there is a high chance of rain. Unsupervised document classification is based on cluster analysis techniques and does not require human interaction. Usually, we use unsupervised learning when we have unlabeled training data, but the assumption is that the data is correlated, and there are some inner relations. Semi-supervised document classification uses both labeled and unlabeled training examples. Semi-supervised learning is a combination of both supervised and unsupervised learning.

Comparison analysis[1] of document classification techniques is summarized in Table 2.1.

Table 2.1: Comparison of document classification approaches

	Expert judgments	Crowdsourcing	Machine learning
Classification	Manual	Manual	Automatic
Quality	Excellent	Good	Noisy
Cost	Very expensive	Moderately expensive	Moderately cheap
Scalability	Do not scale well	Scale to some extent (budget)	Scale very well

2.2 Feature-based classifiers

Feature-based classifiers use features in supervised learning. Most popular learning algorithms are:

- Support Vector Machines (SVM)
- Linear regression

[1] K. Balog. Retrieval Evaluation. DAT630.

- Logistic regression
- Naive Bayes
- Linear discriminant analysis
- Decision trees
- K-nearest neighbor algorithm

Some neural network models use manual features in the learning process as well. On average, features based algorithms are effective on small datasets, where data correlation and data relation can be expressed in a definite number of manually designed features.

Many algorithms had been originally designed for binary classification problems. In a binary classification problem, we are deciding between only two classes. For example, we decide either the person is female or male, or we check if the person is eligible for getting a tax return or insurance payment. Another example is the game outcome: win or loss. Binary classification problems can be answered with simple, "Yes" or "No" statements.

Most real-life situations tend to solve multi-class problems. By multi-class classification problem, we understand the problem with more than two possible outcomes. For example, by provided ingredients predict the meal: breakfast, lunch or dinner. Another example is to identify a single written digit from 0 to 9, ten classes in total. In order to solve the multi-classification problem predictions should be combined. There are two known approaches to solve a multi-classification problem: one-against-one and one-against-rest.

In one-against-one approach, we construct the binary classifier for each pair of classes in a class set ($\frac{k(k-1)}{2}$ binary classifiers in total). The positive class will receive a vote in each pairwise comparison. As a result, target class is a class with a majority of votes.

In one-against-rest approach, we have a set of possible outcomes (classes). For each class, instances which belong to specific class y_i are positive examples. All other classes in a set are treated as negative examples. If an instance has been classified as positive, the positive class gets a vote. Otherwise, all classes except for the positive class receive a vote. The class with a majority of votes is being selected and returned.

Support Vector Machines (SVM) Support Vector Machine is a machine learning model for data classification. SVM works best with binary classification tasks, however, it can be used to solve multi-class problems. The goal of SVM is to find (design) the hyperplane which separates two classes with a maximum margin. It is possible to find multiple hyperplanes for the same training data, however, the best hyperplane has the maximum margin from both classes. The class examples closest (have minimum distance)

to the hyperplane, are called support vectors. Only support vectors are important, other data points can be ignored. The Figure 2.1 illustrates the principle of SVM. In order to define a hyperplane, classes should have some qualities or numerical presentations called features. In the document classification problem, SVM uses TF-IDF vectors as features.

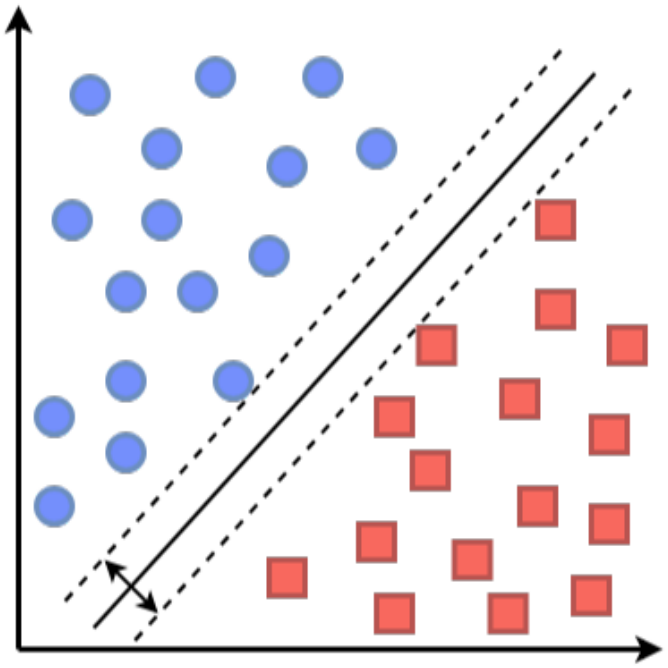


Figure 2.1: State Vector Machines logic

Support vectors are used to define the hyperplane equation. The hyperplane is used to classify data later on. Non-linear functions, such as; polynomial, radial basis function (RBF), or sigmoid; help to find the hyperplane in the non-linear space. These functions are known as kernels.

SVMs had proven the ability to handle large feature spaces without feature selections. Support Vector Machine also tends to be robust to over-fitting [2].

2.3 Neural networks and deep learning

Neural networks are often denoted as an Artificial Neural Networks is an Artificial Intelligence paradigm inspired by Neuroscience; science about human brain functioning [1]. The human brain can efficiently identify patterns and solve complicated tasks. Scientists had an idea of transferring human brain functionality, to machines, to create smart computers. Inspired by Neuroscience researches and scientists, have been mapped neurons in the human brain to input, hidden and output units, and connections between units to synapses. Neurons are connected with each other, their job is to pass information further.

They aggregate all synapses and apply activation function. Multiple neurons form input, hidden, or output layers. Activations in one layer determine activations in the next layer.

Neuroscientists still do not know completely how the human brain works. For that reason, Artificial Intelligence science had to find other sources of inspirations, but the structure and basic idea of the human brain find reflection in the neural network architectures.

There are three types of units in neural networks: input units, hidden units, and output units [1]. Groups of units form input layer, hidden layer, and an output layer. Networks with more than one hidden layer are called deep neural networks. Deep neural networks tend to identify more complicated patterns comparing to neural networks. The amount of hidden layers is not limited, and there are no guidelines on how many hidden layers should be used. The general structure of Deep Neural Network is illustrated in Figure 2.2.

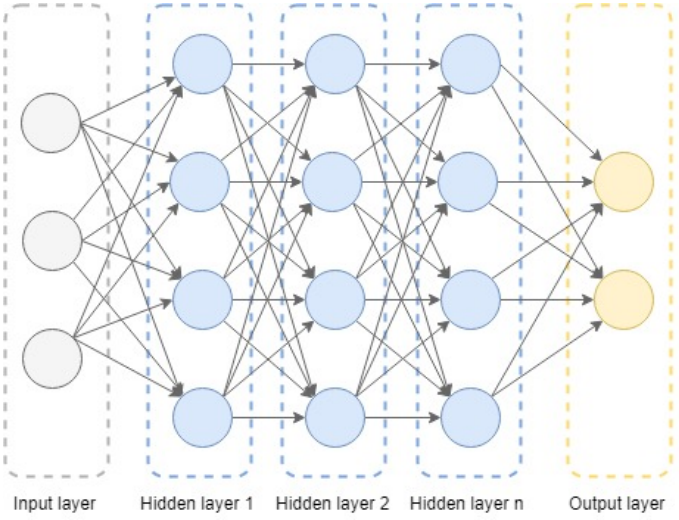


Figure 2.2: General structure of Deep Neural Network

Units are connected with other units via weights. Weights are represented by weight matrix W. Weight matrix determines how features affect the prediction. There is one additional parameter b, called bias. Bias is an intercept term which reflects how close is the estimate to the true value [1]. Hidden units are responsible for processing incoming data. Hidden units take a weighted sum of inputs along with biases. The activation function is applied on top of hidden units to determine the output which is passed further to the next layer. There is a variety of activation functions.

The cost function analyzes the output predictions of a neural network and tells how good the network performs. The cost function provides the feedback which is used to optimize the weights. The model computes series of derivatives to compute the gradient and updates the weights and biases. This process is called back-propagation.

DNNs perform operations on tensors. Tensors are multi-dimensional data structures which can be thought of as a generalized matrix. Training of DNN involves weights optimization, and loss minimization during back-propagation. Back-propagation DNN operations include linear operations and non-linear operations such as Tanh or ReLu (rectified linear units). DNNs have many architectures and hyperparameters. Theoretically, DNN can approximate any function [3]. DNNs are data-hungry and need large-scale training data corpus [4].

Trained neural network models are expected to perform well on unseen data. Since text representations commonly are learned from training set distribution, testing the model on a new corpus with a different distribution, may lead to poor model evaluation and performance. Different corpora have specific patterns learned by a neural network. DNNs suffer from corpus variance problem, this is also known in the literature as adversarial examples [1]. For example, we have a model which identifies handwritten digits from 0 to 9 with 98% accuracy. After applying, almost invisible for a human eye noise to the images in the testing set, the accuracy of the model can drop to 10-15%. Generative adversarial networks [5] are called to deal with adversarial examples. To address the problem of different corpora distributions model can be retrained on different datasets.

The neural model should be able to handle inputs of various length, since news articles have different lengths. Some articles are 750 characters long, other articles are 3000 characters long. Not all content is equally meaningful, sometimes one section or paragraph expresses the meaning of the whole article (document). Training data is always noisy and contains errors. The model should still learn good representations from noisy inputs. Another approach is to ignore error inputs.

Challenges (or good model must handle) [4]:

1. Semantic understanding
2. Robustness to rare inputs
3. Robustness to corpus variance
4. Robustness to variable lengths inputs
5. Robustness to errors in the input
6. Sensitivity to contexts

The main advantage of neural networks and deep neural networks is a feature free design. These networks do not need manually designed features to identify patterns, estimate functions and provide accurate predictions. The smart neural network architecture is a

key to success. We can consider each neural network as a building block. Those building blocks can be used to build different models. The combination of different network architectures can provide unique solutions.

2.4 Feedforward neural networks

Feedforward network is a neural network architecture without loops (cycles). Feedforward networks support a single flow of information: from the input to output [1]. Feedforward network classifier general formula:

$$y_i = g(\sum_j W_{ij} x_j + b_i)$$

Each hidden unit h takes input x , multiplies it by weights W, sums it up with bias b and passes through the activation function g to yield the output y. Feedforward networks work with fixed size inputs. It is also assumed that training examples are independent. The general structure of feedforward network is illustrated in Figure 2.3.

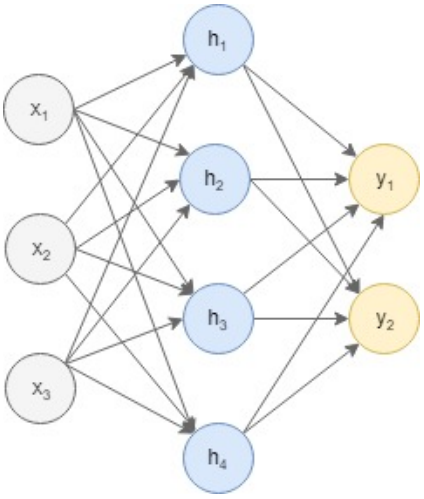


Figure 2.3: Feedforward neural network structure

2.5 Convolutional neural networks

Convolutional neural networks (CNNs) are neural networks designed to handle data with known deterministic grid topology and involve convolution operation. Convolution is a special linear operation on two functions in algebra. One of the functions is called weighting function $w(a)$, known as a kernel, while another one is a regular function which we assume to be noisy, known as an input. By combining these two functions we get

a smoothed estimate. Usually, the kernel is much smaller than input which gives us sparse output. In literature convolution operation is denoted with asterisk ($*$). The output of convolution operation is commutative and is often called the feature map [1]. Convolution neural network structure is shown in Figure 2.4.

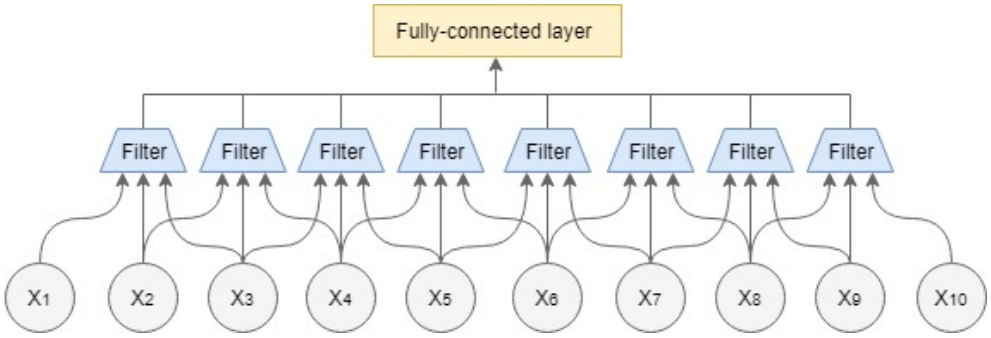


Figure 2.4: Convolutional neural network structure

General convolution formula:

$$s(t) = (x * w)(t) =$$

$$\sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

Where $s(t)$ is a state at the timestamp t, x is the input, w is the weighting function or kernel, a is the age of the measurement.

For single dimension, a slightly different notation formula is:

$$S(i) = (I * K)(i) =$$

$$\sum_{m} I(m)K(i-m)$$

Where $S(i)$ is the state at timestamp i , I is the input and K is the kernel.

For two-dimensional input case:

$$S(i,j) = (I * K)(i,j) =$$

$$\sum_{m}\sum_{n} I(m,n)K(i-m,j-n)$$

Convolutional neural networks are associated with sparse iterations, parameter sharing, and equivariant representations [1].

Sparse iterations

In traditional neural networks, each output unit interacts with each input unit. These

networks are called fully-connected. Due to the sparse representation of convolution output, CNNs are more computational and memory effective.

Parameter sharing
Parameter sharing allows using the same parameters for multiple functions in a model. In CNN each element in the kernel is used at each position of the input.

Equivariant representations
Parameter sharing property in CNN leads to equivariance to translation. Equivariance property refers to immunity to small changes in input. In other words, small changes in input cause small changes in output. If the task of the model is to find some shape on the image despite the location of that shape, the equivariant property becomes very useful.

CNN layer consists of three steps: convolution operation, rectified linear unit (ReLU) activation and finally pooling. Specified steps are not bounded to one specific layer, each step can be a separate layer in a model. Convolutions are usually performed in parallel. Feeding linear activation through nonlinear activation function such as ReLU is called detector. Pooling function aims to replace the output with some summary statistics. There is a variety of pooling functions: max pooling, the average of a rectangular neighborhood, the L2 norm of a rectangular neighborhood and the weighted average. Max pooling is the most popular pooling technique. In max pooling, we select and return the maximal value of a rectangular neighborhood.

CNNs can be used to produce low-dimensional, structured object representations. We know that convolutional networks produce a shrunken output which has lower dimensionality compared to an input.

CNN benefits: processes inputs of variable size, fast, easy on memory, robust to translations in the input, uses fewer computations and demonstrates statistical effectiveness. CNN disadvantages: does not remember the previous state (does not have memory).

2.6 Recurrent neural networks

Recurrent neural networks (RNNs) were designed for learning (remembering) sequences: a sequence of pictures to represent video, a sequence of characters or terms (words) to represent a document. RNN remembers previous events and pays attention to important parts of the past. Unlike feedforward networks, hidden units in RNN depend not only on the input but also on the output of the previous timestamp multiplied by a recurrent network weight matrix. Unrolled in time RNN forms a feedforward neural network.

Chain-rule is used to back-propagate the gradients. The recurrent network structure is illustrated in Figure 2.5.

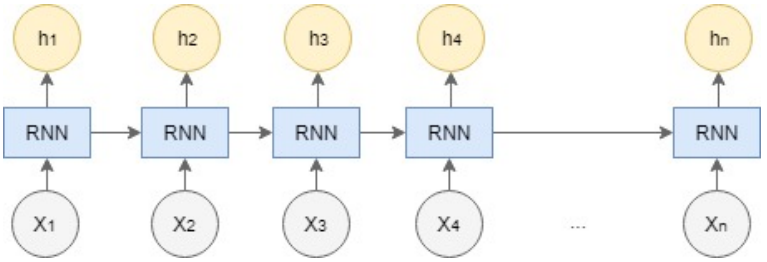


Figure 2.5: Recurrent neural network structure

Hidden unit calculations:

$$h^{(t)} = g_h(W_I x^{(t)} + W_R h^{(t-1)} + b_h)$$

The output is calculated by the formula:

$$y^{(t)} = g_y(W_y h^{(t)} + b_y)$$

RNN has different architectures depending on a number of inputs and outputs: one to one, one to many, many to one and many to many variations. One to one solves categorization problems, given static input get a category or classify it. One to many describes the input with multiple outputs, for example, we feed the image as input and expect to get a textual description (caption) of objects on the image. Many to one is used to recognize an action, get sentiment from a sequence of text. Many to many architectures are used for video summarization and language translation tasks. RNNs are used for sequence generation, text generation, stock prediction, voice recognition, language translation, etc.

RNN benefits: works with variable length input sequences and remembers its state. RNN disadvantages: struggles from long-term dependencies (fail to remember long sequences). Since weight matrix of RNN is shared across all the layers, we have back-propagate to the very beginning of the sequence. This causes vanishing gradient and exploding gradients problems. In vanishing gradient problem, weight product of small weights decreases rapidly and network loses the ability to remember. In exploding gradient problem, weight product of large weights increases exponentially causing learning process to be unstable.

Possible solutions to vanishing/exploding gradient problems include clipping gradients at the threshold, using adaptive learning rate algorithms, applying ReLU activation function or using another recurrent neural network architectures such as Long short-term memory (LSTM) and Gated Recurrent Units (GRUs) [1].

Chapter 3

Related Work

News articles are strings of characters which have to be transformed into a representation suitable for the learning network. Machine learning algorithms and neural networks do not process raw text input. In news background linking task, we work with news articles and news stories contents. In Section 3.1, we discuss local, distributed text representations and pre-trained embeddings. Background linking task involves retrievement of relative (similar) news articles/stories. In Section 3.2, we mention several notions of similarity which our model should detect and, in Section 3.3, go through popular similarity metrics.

We notice some similarities between background linking and information retrieval tasks. In information retrieval field given a query, we should select relevant documents. In news background linking task given a news article, we should retrieve relevant news articles (stories). In Section 3.4, we review traditional IR models, while in Section 3.5, we talk about existing neural approaches.

3.1 Raw text representations in Machine Learning

Neural networks as any other machine learning algorithm cannot process raw text input. Raw text should be represented in a numerical form to perform mathematical calculations and manipulations. Typically, the smallest meaningful unit in machine learning is a term (single word). Different vector representations generalize data in different ways focusing either on distinct terms or common attributes. Since data is represented as vector, simple vector math operations can be applied to it. Each term can be expressed using local or distributed representations. There are several ways to represent the news article content:

1. Manually design features which can represent a news article.

2. Use a news article one-hot encoded term vector.

3. Generate a news article embedding.

3.1.1 Local representations

Usually, one-hot encoded vectors are used to generate a local representation of term. One-hot encoded vectors are binary vectors used to represent objects in a fixed size vocabulary. The one-hot encoded vector contains only one 1 value, the rest of the values are zeros. Each position of 1 corresponds to a unique term.

For example, assume we have a dataset which describes meals of the day V = ("breakfast", "lunch", "dinner"), where V is a vocabulary represented as a set of values. By marking the value as one, we give it a notion of presence. Hence, local representation for "breakfast" is [1, 0, 0], for "lunch" - [0, 1, 0] and for dinner - [0, 0, 1]. The graphical representation is shown in Figure 3.1. Each term as the unique entity is highlighted with green color. Each term outside of the vocabulary has no representation or denoted with special "UNK" symbol [4].

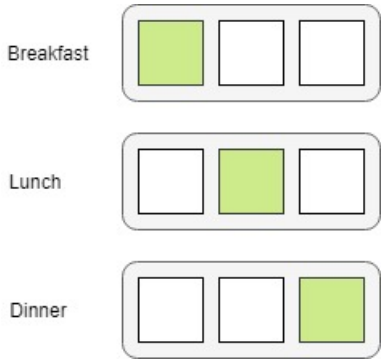


Figure 3.1: Local representations for meals of the day

3.1.2 Distributed representations

In distributed representation, each term is represented by a dense or sparse vector of its attributes. Distributed representation is a vector of hand-crafted features or a learned representation in which the individual dimensions are not interpretable in isolation [4]. Each term in distributed representation has a list of properties (attributes). Presence or absence of those properties describes the term.

Distributed vector representations can be retrieved by aggregating and combining local representations. For example, we have a vocabulary with dishes: V = ("oatmeal",

"scrambled eggs", "sandwich", "salad", "steak"). Local representations of terms in the dishes vocabulary are illustrated in Figure 3.2.

Figure 3.2: Local representations for dishes of the day

Combination of two local can give us distributed representation for all terms, check Figure 3.3. Oatmeal and scrambled eggs are two different dishes, however, they share one common property: both dishes are usually served for breakfast.

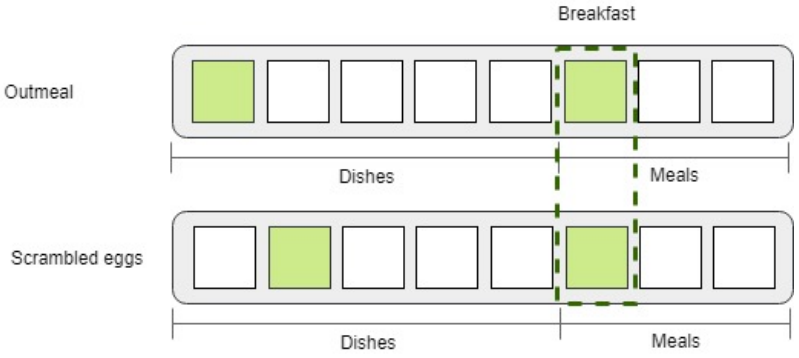


Figure 3.3: Distributed representations for dishes of the day

Term vectors can be represented as a space model in high dimensional space. The example in Figure 3.4 shows that Term 1 and Term 4 are closer in space to each other, this indicates that they are more relevant to each other.

3.1.3 Embeddings

Embeddings are low-dimensional dense vector representations. Term embeddings show good results term analogy tasks. Explicit (distributed) vector representations are based on distributional features. Explicit vectors are sparse and high dimensional. The number of dimensions depends on the number of documents or vocabulary size. Sparse vector representations are not practical for big data corpora. Embeddings represent data in lower dimensional space and preserve properties of data and relationships between

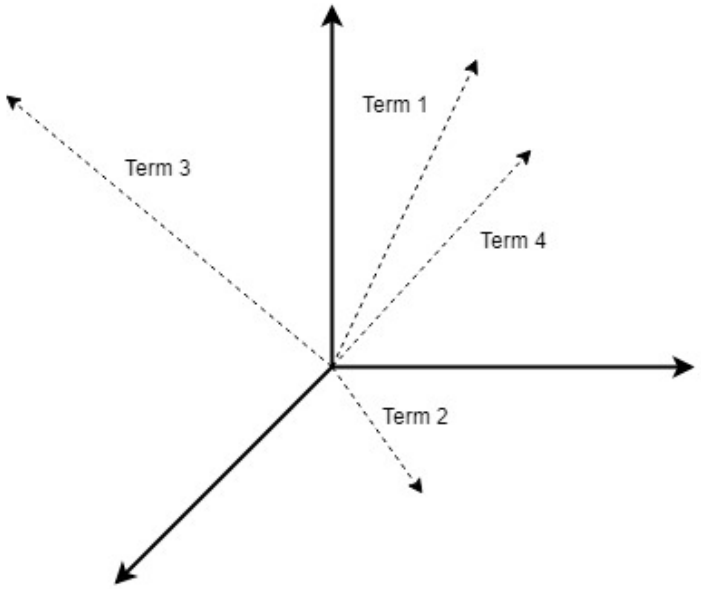


Figure 3.4: Vector space model

terms. Embedding can be retrieved from explicit vectors. Terms in embeddings are more generalized and more practical.

Embeddings can be retrieved from neural and non-neural computations.

Latent Semantic Analysis (LSA) builds embeddings by performing singular value decomposition (SVD) over term-document matrix where rows correspond to terms and columns to documents. LSA is a part of global matrix factorization methods. These methods decompose large matrices to capture statistical information about the corpus. LSA groups different terms that occur in a similar context into the same semantic cluster, hence, low-dimensional representations of documents can have high similarity without sharing common terms. LSA effectively leverages statistical information but performs poorly on word analogy task [4, 6].

In **word2vec**, term features are extracted by sliding fixed size window over term neighbors in a text of training corpus. Word2vec implements the skip-gram model which is a one hidden layer neural network based on the product of two term vectors. Word2vec generates IN embedding which corresponds to all input terms and OUT embedding which corresponds to output terms. Only IN embeddings are used, OUT embeddings are discarded after training. Word2vec performs well on word analogy task but does not take advantage of full utilization of global statistics from the corpus due to training on local context windows. Hence, word2vec does not capture repetitions in the data [4, 6].

GloVe (GlobalVectors) also implements the skip-gram model and is trained on individual term-neighbor pairs. The GloVe also has IN and OUT embeddings but sums

them up to retrieve a term embedding [4]. The GloVe combines advantages of LSA and word2vec approaches. The GloVe is trained on global term-term co-occurrence counts. The GloVe performs well on word analogy task. Word analogy task answers questions like: "x to y as z to ___?" The goal of word analogy task is to fill the gap with the most appropriate word. On the mathematical side, we are looking at word representations w and trying to find word representation which is the closest to $w_y - w_x + w_z$ by means of cosine similarity measure [6].

We will demonstrate some word analogy examples retrieved by performing vector operations on the GloVe term embeddings. We will show top 10 results for each equation and use GloVe 6B 300 dimensional term representations[1].

Standard "king" - "man" + "woman" equation results are shown in Figure 3.5. Man to woman as king to queen.

```
> king - man + woman

queen                0.31
monarch              0.44
throne               0.44
princess             0.45
mother               0.49
daughter             0.49
kingdom              0.50
prince               0.50
elizabeth            0.51
wife                 0.52
```

Figure 3.5: Word analogy for equation: "king" - "man" + "woman"

Another word analogy example for equation "football" - "field" + "ice" results are shown in Figure 3.6. Football is associated with a field, while hockey is associated with ice.

```
> football - field + ice

hockey               0.34
basketball           0.49
soccer               0.50
league               0.56
footballer           0.56
nhl                  0.57
rugby                0.57
club                 0.57
skating              0.57
under-18             0.57
```

Figure 3.6: Word analogy for equation: "football" - "field" + "ice"

GloVe pre-trained term embeddings show good results on word analogy task. For some, not common term vector equations we got the following results. The example, "engineer" + "developer" results are shown in Figure 3.7. The output vector indicates that person with engineering and software development skills is very likely to be an entrepreneur.

[1] https://github.com/brannondorsey/GloVe-experiments

```
> engineer + developer

entrepreneur          0.40
architect             0.43
builder               0.43
contractor            0.47
inventor              0.47
programmer            0.47
businessman           0.48
engineers             0.50
engineering           0.50
consultant            0.50
```

Figure 3.7: Word analogy for equation: “engineer” + “developer”

Term embeddings are good at deriving summary term vectors out of the content. For example, “infant” - “crying” results are shown in Figure 3.8. It is clear that when the newborn infant is not crying, then there is a high chance that there is something wrong with the baby.

```
> infant - crying

mortality             0.58
maternal              0.61
infants               0.63
newborn               0.63
birth                 0.66
incidence             0.68
expectancy            0.69
immunization          0.70
perinatal             0.70
births                0.71
```

Figure 3.8: Word analogy for equation: “infant” - “crying”

Some terms cause confusions and protest in human minds. The “love” - “feelings” equation results are shown in Figure 3.9. Usually love is associated with warm feelings and attraction. Love without feelings is rather a contradiction and causes the note of protest.

```
> love - feelings

!                     0.64
starring              0.67
starred               0.68
classics              0.68
lovers                0.69
featuring             0.69
loves                 0.69
classic               0.70
comedy                0.70
beautiful             0.70
```

Figure 3.9: Word analogy for equation: “love” - “feelings”

Paragraph2vec is similar to word2vec and is used to create an embedding for paragraphs [4].

LSA, word2vec, and GloVe term embeddings are popular and widely used. LSA and paragraph2vec capture topical similarities while word2vec and GloVe capture both: topical and typical notions of similarity.

3.2 Notions of similarity

In semantic representation the evidence of aboutness is important. Aboutness links terms to contextually related terms. For example, pumpkins, costumes, and candies are relative to Halloween. Semantic representations should also capture the synonyms of terms; "nice" and "good", "man" and "boy' are similar. Latent representation of intent is important [4]. The model should be robust to rare inputs. Most learned representations are based on limited vocabulary. Hence, the poorly designed model will fail to perform well on unseen terms.

Notions of similarity:

1. Typical
   Terms are considered similar if they share a common attribute or property type. For example, Norway and Spain are both countries. The typical similarity is more likely to map Norway to Spain than Norway to Vikings.
2. Topical
   Terms are considered similar if they are related to some common topic. For example, Norway is more similar to Vikings, Fjords and, Trolls than to the USA.
3. Linguistic Considers the linguistic style of a document. For example, news articles style is different from government reports or lawsuits.

A good model should be able to catch multiple notions of similarity.

3.3 Similarity metrics

To measure the distance between objects in space different similarity metrics are used. Similarity metrics used to calculate proximity between two objects have following properties [7, 8]:

1. $sim(x, y) = 1$ only if $x = y$ $(0 \leq sim \leq 1)$
2. $sim(x, y) = sim(y, x)$ for $\forall x, y$ (symmetry)

Prior to using similarity metrics, some actions are required. Prerequisites:

1. Data should be normalized. For example, min-max normalization (feature scaling) can be used [2].
2. Data should be binarized if similarity metric works only with binary values.

3.3.1 Euclidean Distance

The output of Euclidean distance is a distance between two data objects shorter distance indicates higher similarity [7]. Euclidean distance is calculated by the formula below:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} =$$

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

3.3.2 Pearson Coefficient

Pearson Coefficient computes similarity by drawing a line between attributes of two objects. Correlation between two objects results in positive slope line. Pearson Coefficient is more robust to unnormalized data [7]. Pearson Coefficient is calculated by the formula below:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} =$$

$$\frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

3.3.3 Jaccard Coefficient

Jaccard Coefficient works with objects with binary attributes and is based on calculating the proportion of total matching elements versus the sum of partly matching and a total matching [7, 8]. Jaccard Coefficient is a good metric to work with sparse vectors. Jaccard Coefficient is calculated by the formula below:

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

where: f_{11} indicates that x = 1 and y = 1, f_{01} - x = 0 and y = 1, f_{10} - x = 1 and y = 0.

[2] https://en.wikipedia.org/wiki/Feature_scaling

3.3.4 Cosine similarity

Cosine similarity is measured as the angle between two vectors [7, 8]. Commonly used to find similarity between text documents. Effective similarity metric in semantic space. Cosine similarity is calculated by the formula below:

$$cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

where $A \cdot B = \sum_{i=1}^{n} x_i y_i$ is a vector dot product, $||X||$ and $||Y||$ are lengths of vectors X and Y.

3.3.5 Tanimoto Coefficient (Extended Jaccard Coefficient)

Tanimoto Coefficient measures the similarity between document data. If documents have binary representations Tanimoto Coefficient turns into Jaccard Coefficient [7]. Tanimoto Coefficient is calculated by the formula below:

$$T(X, Y) = \frac{X \cdot Y}{||X||^2 + ||Y||^2 - X \cdot Y}$$

3.4 Traditional models in IR

3.4.1 TF-IDF and BM-25

Term Frequency-Inverse Document Frequency (TF-IDF) represents a proportion of a number of occurrences of each term in one document and document collection overall. BM-25 is a probabilistic ranking function built on of TF-IDF. The formula:

$$BM25(q, d) = \sum_{t_q \in q} IDF(t_q) \cdot \frac{TF(t_q, d) \cdot (k_1 + 1)}{TF(t_q, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$

Where $avgdl$ is average document length in the collection D, $k1$ and b are parameters that should be tuned. Defaults value for $k1 = [1.2, 2.0]$ and $b = 0.75$. The IDF is computed by the formula:

$$IDF(t) = log \frac{|D| - df(t) + 0.5}{df(t) + 0.5}$$

Where df is a document frequency.

BM25 only considers contributions of individual terms, hence, implies direct term matching approach.

3.4.2 Language modeling

Language modeling (LM) is based on posterior probability $P(D|Q)$.

TF-IDF based approaches and LM are based on terms count and do not catch positional and relational connections within terms and content.

3.4.3 Learning to rank

Learning to rank (L2R) train models over a set of hand-crafted (manually selected) features. Traditional information retirement model is Latent Semantic Analysis (LSA). LSA learns dense vector representations of terms and documents. RankNet is a pairwise loss function [4].

3.5 Existing Neural Approaches

Neural networks work with character-level or term-level text data.

In character-level, each character is represented as the one-hot encoded vector. Length of the one-hot encoded vector is equal to the length of the vocabulary. Dimensions are also known as channels. Sentences, paragraphs, and texts can be represented by combining (aggregating) character vectors. Character level vector representations have no prior knowledge about language specific and learn patterns.

One of the simple approaches used in machine learning is mapping unique characters in total vocabulary to integer numbers. For example, our vocabulary consists of "Hello world!" phrase. We ignore case sensitive structure and treat all characters as lower case. After tokenizing the vocabulary we get following unique characters: voc = [h, e, l, o, w, r, d, !], where len(voc) = 8. In second step we map characters to integers: $charToInt = \{h:1, e:2, l:3, o:4, w:5, r:6, d:7, !:8\}$ and $intToChar = \{1:h, 2:e, 3:l, 4:o, 5:w, 6:r, 7:d, 8:!\}$ for fast backward conversion. On the output, we have the phrase "Hello world!" represented as the vector= <1 2 3 3 4 5 4 6 3 7 8>. Finally, integers should be converted to one-hot encoding. Letter 'h' for instance has one-hot vector <1 0 0 0 0 0 0 0>. The current way of converting text to integers only valid for models which do not retrieve any meaning of the word from the context. Before-mentioned set up can be used for predicting text sequences with RNN LSTM[3].

[3] https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

To retrieve term-level text data (words) raw text is tokenized into terms. Each vector is then represented as sparse (explicit vector representation) or dense (pre-trained word embeddings) vector. Explicit vector representation can be retrieved from local representation, aggregated local representations or distributed representations. Aggregated term representations can form sentence, paragraph or document representation.

Popular DNN architectures:

- ***Input-invariant models***
  CNNs and RNNs are used as input-invariant models because they are robust to input-invariants. Such networks catch words and language meaning independently from the position of their occurrence in the document [4]. These architectures have fixed size windows which slide across input content with some fixed step. The filter (kernel function) is applied to each window position with shared parameters to extract some features and patterns. The filter is also denoted as a cell in machine learning literature [1].

  In CNN architecture each cell is multiplied by the weight matrix. Pooling operation is applied to cell output to aggregate some features. Global pooling operation across all aggregated features produces fixed size output (fully-connected layer - dense vector representation). In CNNs convolution and pooling operations are performed within independent (unique) window. In RNN output of the previous cell is also considered. LSTM is popular RNN architecture with 4 gates which control what is kept into the memory and what should be forgotten [1].

- ***Auto-encoders***
  Auto-encoders are neural network architectures which tend to learn compressed representations of objects out of their high dimensional representations. Compressed representations are later decoded to approximate initial input. Model is trained to minimize the difference between input and output. Good auto-encoders can reproduce initial input with minor changes. Compressed representations of objects can be used as training data for other neural network architectures. Auto-encoders are used to create summaries of text documents, rewrite comments and reviews. There is a trade-off between the level of compression and a quality of decoded data.

- ***Siamese networks***
  Siamese networks were initially designed for comparing signatures and fingerprints [4]. Later Siamese networks had been adjusted to work with short texts. Siamese network consists of one model which performs the auto-encoder function - retrieve the compressed representation of short texts. Siamese networks are trained on pairs of similar and not similar inputs. Usually, Siamese networks share the

same model for both inputs. Pairs of input are fed into the designed model to retrieve dense vector representations. Vector representations are fed into cosine similarity function to measure relativeness. The goal is to optimize parameters of the model in such a way that similar objects are closer to each other in semantic space and not similar object have larger distance. The model has auto-encoder architecture is input representation is compressed, otherwise other architectures are used.

Deep Semantic Similarity Model (DSSM) is a Siamese network used in IR to retrieve relevant document titles by short queries. The DSSM consists of two models: one for query, another for document title. Cosine similarity is used as a similarity metric between low-dimension vector representations [9].

The DSSM is widely used for short texts, our goal is to adapt DSSM for background article linking task.

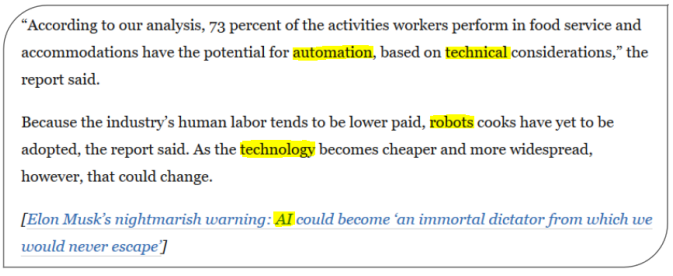
Chapter 4

Solution Approach

In this chapter, we introduce our solution approach. In Section 4.1, we talk about news background linking task. In Section 4.2, we analyze similarity between information retrieval and background linking tasks. Finally, in Section 4.3, we introduce Deep Semantic Similarity Model designed for solving background linking task.

4.1 Introduction

The goal is given a news articles retrieve related (similar) news stories. Real life example is illustrated in Figure 4.1. We manually highlighted some keywords which contributed to the current selection choice. Keywords *"automation"*, *"technical"*, *"robots"* and *"technology"* led to linking of an article within *"AI"* topic.


"According to our analysis, 73 percent of the activities workers perform in food service and accommodations have the potential for automation, based on technical considerations," the report said.

Because the industry's human labor tends to be lower paid, robots cooks have yet to be adopted, the report said. As the technology becomes cheaper and more widespread, however, that could change.

[Elon Musk's nightmarish warning: AI could become 'an immortal dictator from which we would never escape']

Figure 4.1: Background linking example from The Washington Post

Our model should be able to produce a list of candidate news stories with similarity scores. News stories with highest similarity scores are considered relevant to the background context of the news article. The high-level idea of what the model should do is illustrated in Figure 4.2.

Figure 4.2: Background linking task high-level idea

4.2 Analysis

The problem we are trying to solve is close to IR task when given a short query relevant documents should be retrieved. In our case query is a news article and relevant news stories should be retrieved. Neural approaches to solving IR task described in Section 3.5, work with short text strings. Indeed, convolutional neural networks are able to extract key topics and key phrases as features from long texts as well. CNN can produce a high-level dense representation of news articles and news stories. The relevance of two articles can be estimated via proximity measures. The idea is to generate news article representation which catches the distribution over information in it. The high-level is shown in Figure 4.3.

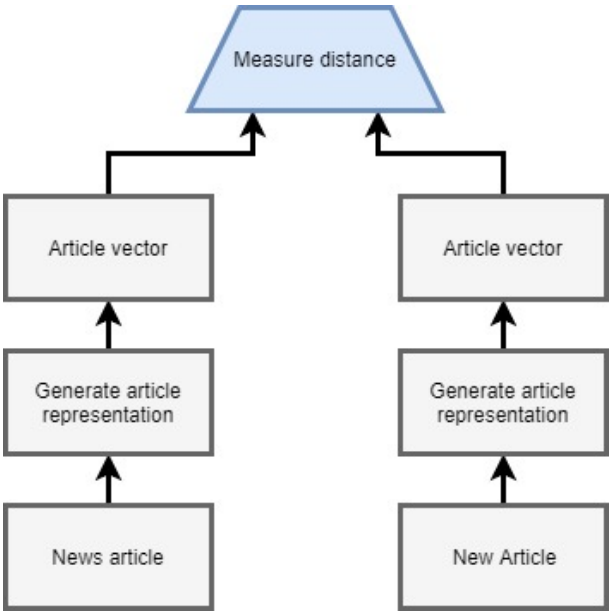


Figure 4.3: The high-level solution idea

4.3 Proposed Solution

4.3.1 Deep Semantic Similarity Model

Deep Semantic Similarity Models also called Deep-Structured Semantic Models denoted as DSSM. Deep Semantic Similarity Models had been proposed in 2013 in [9]. Researches proposed DSSM to solve information retrieval problem: map queries to relevant documents. Since 2013 Deep Semantic Similarity Models had been developed and adjusted to the variety of tasks such as: web search [10], information retrieval [11, 12], question answering [13, 14], modeling interestingness [15], automatic textual image descriptions [16], natural language processing [17], machine translation, etc. Different tasks solved by Deep Semantic Similarity Models are summarized in Table 4.1.

Table 4.1: DSSM tasks

Task	X	Y
Web search	Query	Web document
Automatic highlighting	Document	Phrases to highlight
Contextual entity search	Key phrase and context	Entity and corresponding page
Machine translation	Sentence to translate	Translation
Ad selection	Query	Ad keywords
Entity ranking	Mention	Entities
Recommendation	Document	Relevant documents
Nature User Interface	Command (text/speech)	Action
Summarization	Document	Summary
Query rewriting	Query	Rewritten query
Image captioning	Text string	Images

Deep Semantic Similarity Models are called deep because they use Deep Neural Networks to obtain the low-level representations of raw input vectors. As we remember in DNN features are trainable, not manually designed. Various implementations of DSSMs belong to Siamese networks family. In DSSM network we take the input and represent it as an encoding vector (feature vector). The neural architecture which is responsible for generating a representation of textual information does not use softmax function, there is no need to classify anything. Feed different inputs to the same deep neural network to retrieve encoding vectors. Cosine similarity metric is used to measure the similarity between two vectors. If two inputs are the same or very similar cosine similarity score is high. The DSSM original structure is illustrated on in Figure 4.4.

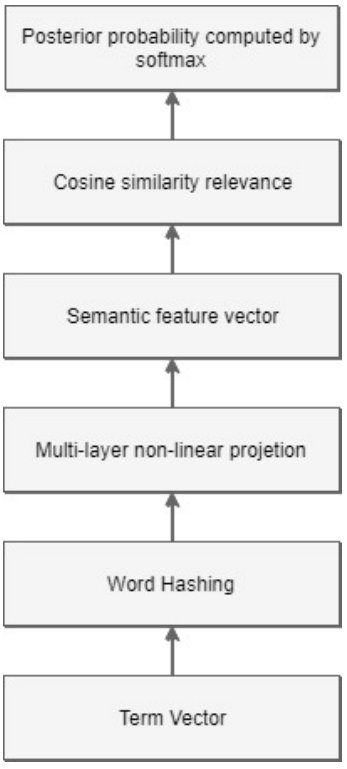


Figure 4.4: DSSM original structure

To learn the representation of news articles and to map news articles and new events we propose using Deep Structured Semantic Models. The DSSM had been used for web search and information retrieval, but to best of our knowledge, no one has applied it to news articles. The relevance of a news article given a news story is computed as a cosine similarity between them. We formulate the task as follows: map news articles to news event (Wikipedia page) to learn news event representation.

Regular neural networks are mainly used for classification with target represented as the one-hot encoded vector. The DSSM is used for ranking, not classification tasks. Target in DSSM is represented as continuous-valued vectors.

General Deep Semantic Similarity Model structure [18]:

1. Convert target from one-hot encoded vector to continuous-valued vector.
2. Compute target vector using Deep Neural Network.
3. Normalize two vectors and compute the distance between them.

The standard DSSM takes word sequence as input, generates term vector representation in convolutional and max-pooling layers and produces abstract semantic vector representations at semantic layer. A convolutional layer is used to extract local features where a max-pooling layer is used to generate global features such as key topics and keywords.

DSSM is learning from labeled X and Y pairs. Lets assume X is a news article and Y^+ - relevant news event (story, Wikipedia page), usually called positive example (pair), Y^- - irrelevant news event, usually called negative example (pair). Y^+ is more relevant than Y^- . Pairwise structure of the model is shown in Figure 4.5.

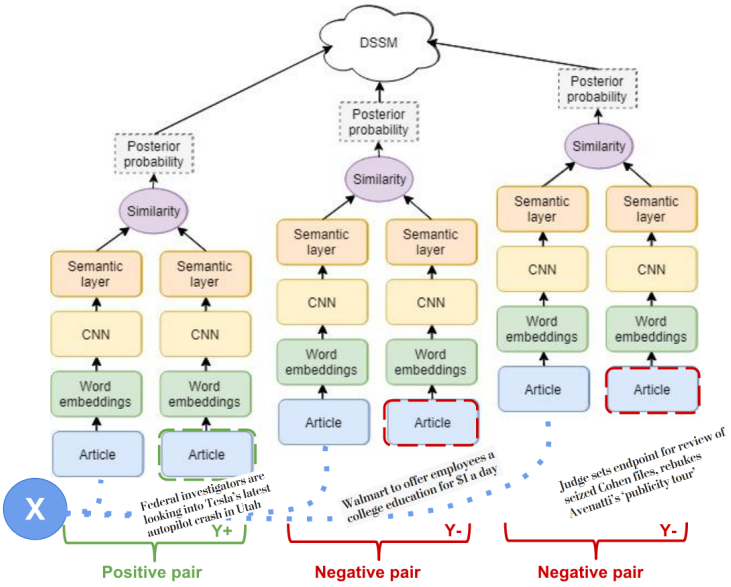


Figure 4.5: The DSSM pairwise structure

$sim\theta(X, Y)$ - cosine similarity of X and Y in the semantic space mapped by DSSM and parametrized by θ. A change of state between positive and negative pairs is calculated by the formula:

$$\Delta = sim_\theta(X, Y^+) - sim_\theta(X, Y^-)$$

Cost function formula:

$$Loss(\Delta, \theta) = log(1 + \exp(-\gamma\Delta))$$

The goal of the standard DSSM is to maximize Δ. The larger the Δ the better representation is learned.

We construct positive pairs from Wikipedia dataset. Negative pairs are either randomly sampled or retrieved via Elasticseach. We describe sampling techniques in detail in Section 5.4. Positive Wikipedia stories are closer to news articles in semantic space than negative Wikipedia pages.

Posterior probability of a news event given a news article from cosine relevance score between them is computed with softmax function calculated by the formula below [9]:

$$P(N_e|N_a) = \frac{\exp(\gamma R(N_a|N_e)}{\sum_{N'_e \epsilon N_e} \exp(\gamma R(N_a|N'_e)}$$

Where γ is a smoothing factor in softmax function, N_e is a set of candidate news events (new stories, Wikipedia pages), N_a is a news article, R is relevance and $N_e^{'}$ is a news event from candidate pairs events. Relevance R is a cosine similarity and in our case is calculated by the formula:

$$R(N_a|N_e) = cosine(V_{N_a}, V_{N_e}) =$$

$$\frac{V_{N_a}^T V_{N_e}}{||V_{N_a}|| \cdot ||V_{N_e}||}$$

Where V_{N_a} and V_{N_e} are semantic vector representations of news articles and news events.

In DSSM parameters are estimated to maximize the likelihood of the relevant news event given a news article. In our case following loss (cost) function should be minimized:

$$L(\wedge) = -\log \prod_{(N_a|N_e^+)} P(N_e^+|N_a)$$

Where $\wedge$ is a set of parameters in neural networks W_i, b_i.

The general structure of the adjusted DSSM model is illustrated in Figure 4.6.

4.3.2 Model input

We use GloVe 6B pre-trained embeddings to represent terms in news articles and Wikipedia news events. GloVe embeddings tend to capture typical and topical notions of similarity [4]. Prior to extracting pre-trained embeddings we tokenize constructed positive and negative pairs and pad sequences which are shorter than manually set minimum content length.

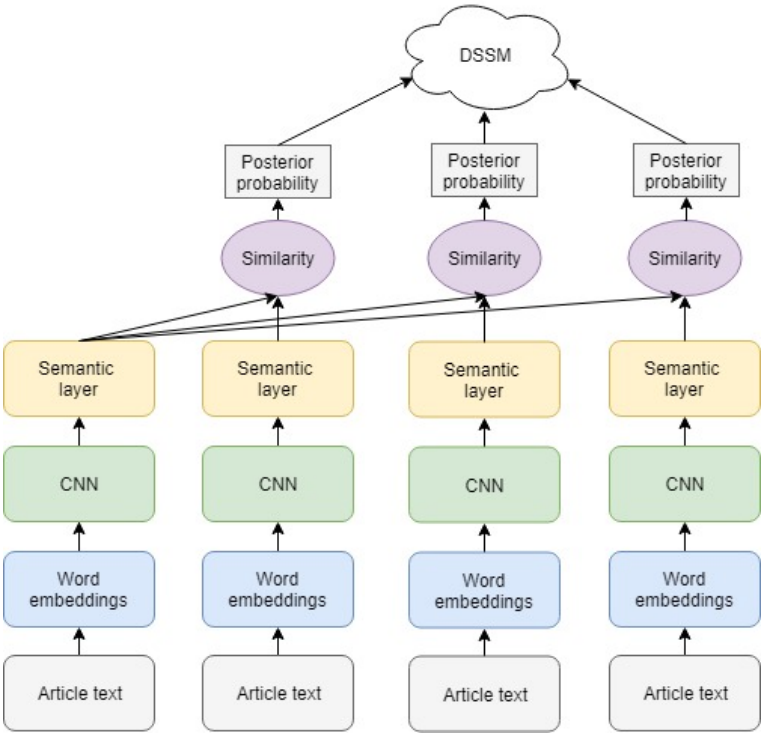


Figure 4.6: The DSSM compact structure

GloVe 6B pre-trained embeddings are trained on Wikipedia 2014 and Gigaword 5 corpora with 6 billion tokens with selected most frequent 400.000 words and symmetric content window size 10 [6]. GloVe 6B contains 50, 100, 200 and 300 dimensional term representations. In other words, each word is represented as a vector with size 50, 100, 200 or 300. During grid search, we did not notice any major differences between the number of dimensions and the accuracy. Indeed, training time grows with the use of higher dimensional representations. We stopped our choice on 100-dimensional representations. For example, the word "truth" in GloVe 6B 50 dimensions is represented by vector in Table 4.2.

Table 4.2: GloVe 6B 50 dimensional embedding of word "truth"

Word/term	Vector representation
truth	0.26273 0.36559 -0.32112 -0.070719 1.3342 0.14024 0.8449 -0.024236 0.12839 0.47655 -0.6505 0.44669 -0.87428 -0.47665 0.76827 -0.0071115 0.52933 0.012882 0.40672 -0.083567 -0.22072 0.97598 0.63861 -0.02576 0.92497 -1.8469 -1.5878 0.21688 0.51778 -0.49787 1.854 -0.59643 -1.0354 -1.0856 -0.73812 -0.49966 -0.14243 -0.60574 0.058185 -0.60367 0.11467 -0.63144 -0.18695 0.49218 -0.39359 0.30666 -0.034556 0.25362 0.22597 -0.34386

4.3.3 CNN architecture

Convolution operation applies a filter to a window of n words to produce a new feature. We apply the filter to all possible windows in the news article text representation to produce a feature map. Max-over time pooling operation over features maps produces the compressed representation of the news article. The convolutional neural network structure is illustrated in Figure 4.7.

Weights in the convolutional network are randomly initialized.

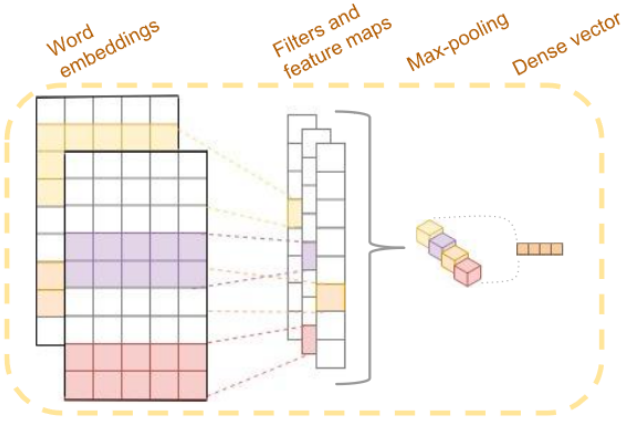


Figure 4.7: CNN with multiple filters

CNN gets pre-trained GloVe 6B 100-dimensional term embeddings on the input. Pre-trained embeddings are good performing feature extractors suitable for various corpora [19].

One filter allows extracting one feature. The idea is to apply multiple filters with different window sizes to extract multiple features. We use Rectified Linear Unit (ReLU), filter windows of size 1, 2 and 3 respectively with 128 filters. Merged features are twice processed with 256 filters with window size 5. The final output of CNN is a dense representation vector of size 256.

The block diagram of adjusted DSSM model with CNN for the case with 3 pairs is shown in Figure 4.8.

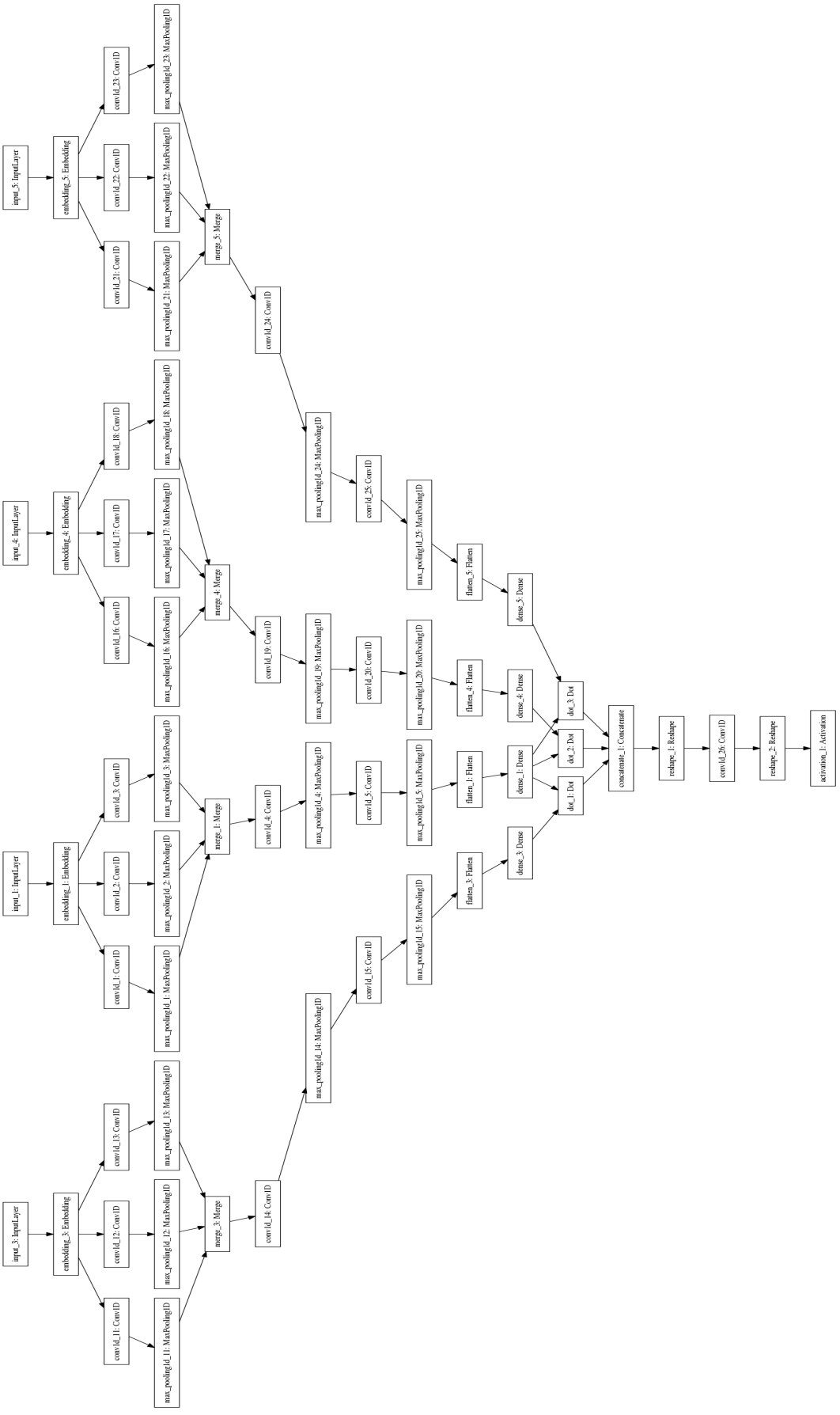


Figure 4.8: Keras model visualization

Chapter 5

Experimental Evaluation

In this chapter, we present the experimental results. We start by introducing datasets in Section 5.1. In Section 5.2, we list the metrics used to evaluate the performance of our model. Experimental setup and model parameters are described in Section 5.3. In Section 5.4, we present experimental results after training the model on Wikipedia dataset with different sampling scenarios. We compare our solution with baseline Support Vector Machine model in Section 5.5. Section 5.6 contains discussions on the topic.

5.1 Datasets

5.1.1 Wikipedia Dataset

Wikipedia extracts dataset consist of two parts: wiki extracts and news articles. Wiki extracts contain 6843 news stories (news events) crawled from Wikipedia. Each entry consists of Title and Content fields, see Table 5.1.

Table 5.1: Wikipedia news events

Field	Definition
Title	Title of Wikipedia page.
Content	Content of wikipedia page. Each section is separated with '==' symbols.

News articles part contains news articles obtained from external source links in Wikipedia news stories. There is 34565 total number of news articles. News articles entry consist of Title, URL, Content, WikiStoryName, WikiStorySectionName, PublicationDate, see Table 5.2. The dataset includes articles majority of which are from 2006 to 2017. 1018

news stories contain at least 10 articles per event, 440 news stories contain at least 20 articles per event.

Table 5.2: News articles from Wikipedia news events

Field	Definition
Title	Title of a news article.
URL	URL of a news article.
Content	Content of a news article.
WikiStoryName	Title of linked Wikipedia page.
WikiStorySectionName	Section name of linked Wikipedia page.
PublicationDate	Publication date of a news article.

The Wikipedia page is a ground truth page for the news event. Wikipedia pages are written manually and evaluated by Wikipedia staff and users. All external sources used in Wikipedia story are referenced and easy to check. The high-level explanation of Wikipedia stories with linked news articles is illustrated in Figure 5.1. Different news articles related to the topic with different chronology are linked to Wikipedia news story about Falcon Heavy test flight.

Figure 5.1: The high-level of Wikipedia stories and linked news articles

In another example, Wikipedia page for 5G[1] describes insights of 5th generation wireless systems. Figure 5.2 shows Wikipedia page structure, where Wikipedia page URL is denoted by 1, Title of the story is denoted by 2, and sections of the story by 3.

[1] https://en.wikipedia.org/wiki/5G

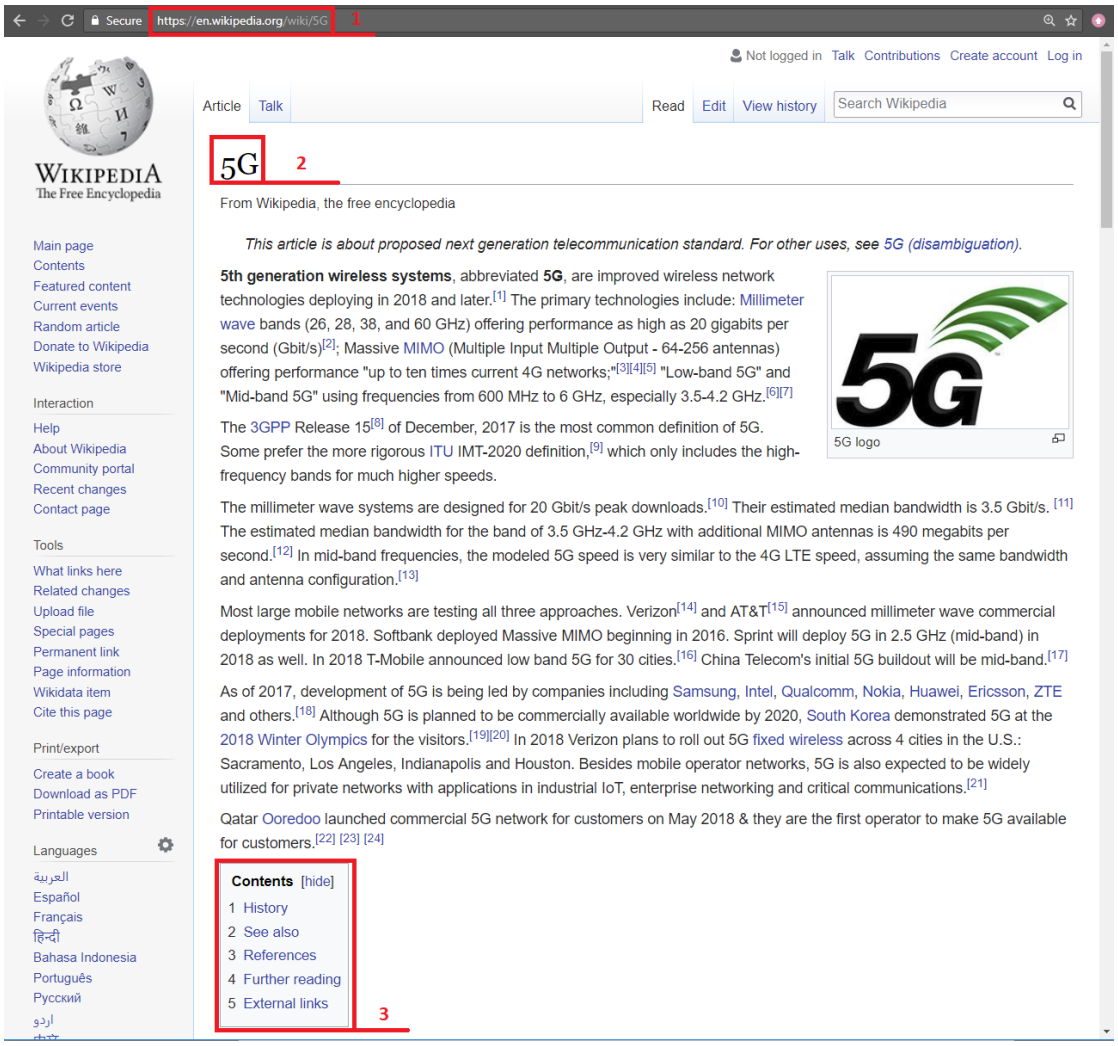


Figure 5.2: Wikipedia page structure

Figure 5.3 shows Wikipedia page references. All references are enumerated, most of them contain active links to view content online.

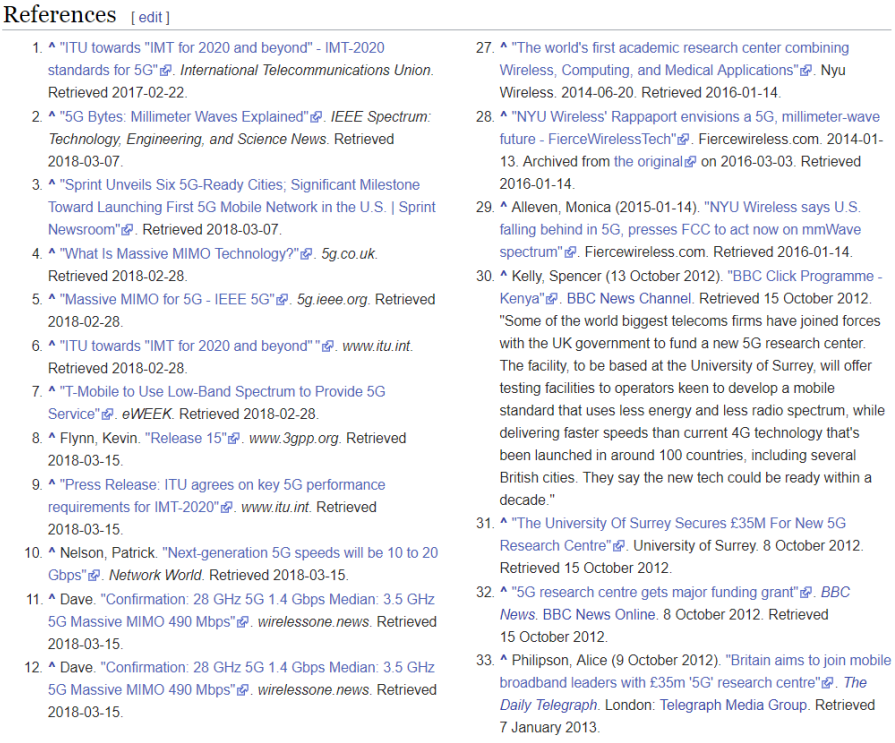
Figures 5.4 and 5.5 show contents of references number 10[2] and 33[3] in Wikipedia reference list 5.3.

Wikipedia has a Current News Portal[4] where current stories are added manually every day from different news media outlets [20]. News stories in Current News Portal are grouped into categories and linked to Wikipedia pages (long-running events).

[2] https://www.networkworld.com/article/2941362/wireless/next-generation-5g-speeds-will-be-10-to-20-gbps.html

[3] https://www.telegraph.co.uk/technology/mobile-phones/9595641/Britain-aims-to-join-mobile-broadband-leaders-with-35m-5G-research-centre.html

[4] https://en.wikipedia.org/wiki/Portal:Current_events



References [edit]

1. ^ "ITU towards "IMT for 2020 and beyond" - IMT-2020 standards for 5G". *International Telecommunications Union*. Retrieved 2017-02-22.
2. ^ "5G Bytes: Millimeter Waves Explained". *IEEE Spectrum: Technology, Engineering, and Science News*. Retrieved 2018-03-07.
3. ^ "Sprint Unveils Six 5G-Ready Cities; Significant Milestone Toward Launching First 5G Mobile Network in the U.S. | Sprint Newsroom". Retrieved 2018-03-07.
4. ^ "What Is Massive MIMO Technology?". *5g.co.uk*. Retrieved 2018-02-28.
5. ^ "Massive MIMO for 5G - IEEE 5G". *5g.ieee.org*. Retrieved 2018-02-28.
6. ^ "ITU towards "IMT for 2020 and beyond" ". *www.itu.int*. Retrieved 2018-02-28.
7. ^ "T-Mobile to Use Low-Band Spectrum to Provide 5G Service". *eWEEK*. Retrieved 2018-02-28.
8. ^ Flynn, Kevin. "Release 15". *www.3gpp.org*. Retrieved 2018-03-15.
9. ^ "Press Release: ITU agrees on key 5G performance requirements for IMT-2020". *www.itu.int*. Retrieved 2018-03-15.
10. ^ Nelson, Patrick. "Next-generation 5G speeds will be 10 to 20 Gbps". *Network World*. Retrieved 2018-03-15.
11. ^ Dave. "Confirmation: 28 GHz 5G 1.4 Gbps Median: 3.5 GHz 5G Massive MIMO 490 Mbps". *wirelessone.news*. Retrieved 2018-03-15.
12. ^ Dave. "Confirmation: 28 GHz 5G 1.4 Gbps Median: 3.5 GHz 5G Massive MIMO 490 Mbps". *wirelessone.news*. Retrieved 2018-03-15.
27. ^ "The world's first academic research center combining Wireless, Computing, and Medical Applications". Nyu Wireless. 2014-06-20. Retrieved 2016-01-14.
28. ^ "NYU Wireless' Rappaport envisions a 5G, millimeter-wave future - FierceWirelessTech". Fiercewireless.com. 2014-01-13. Archived from the original on 2016-03-03. Retrieved 2016-01-14.
29. ^ Alleven, Monica (2015-01-14). "NYU Wireless says U.S. falling behind in 5G, presses FCC to act now on mmWave spectrum". Fiercewireless.com. Retrieved 2016-01-14.
30. ^ Kelly, Spencer (13 October 2012). "BBC Click Programme - Kenya". BBC News Channel. Retrieved 15 October 2012. "Some of the world biggest telecoms firms have joined forces with the UK government to fund a new 5G research center. The facility, to be based at the University of Surrey, will offer testing facilities to operators keen to develop a mobile standard that uses less energy and less radio spectrum, while delivering faster speeds than current 4G technology that's been launched in around 100 countries, including several British cities. They say the new tech could be ready within a decade."
31. ^ "The University Of Surrey Secures £35M For New 5G Research Centre". University of Surrey. 8 October 2012. Retrieved 15 October 2012.
32. ^ "5G research centre gets major funding grant". *BBC News*. BBC News Online. 8 October 2012. Retrieved 15 October 2012.
33. ^ Philipson, Alice (9 October 2012). "Britain aims to join mobile broadband leaders with £35m '5G' research centre". *The Daily Telegraph*. London: Telegraph Media Group. Retrieved 7 January 2013.

Figure 5.3: Wikipedia page external references

Secure | https://www.networkworld.com/article/2941362/wireless/next-generation-5g-speeds-will-be-10-to-20-gbps.html

Data Center Resources/White Papers

NETWORKWORLD FROM IDG

INTERNET OF THINGS SUBNET An independent Internet of Things community View more

Home > Networking > Wi-Fi

DISRUPTOR

By Patrick Nelson, Network World | JUN 29, 2015 8:29 AM PT

About | Thought-provoking commentary on technologies that are changing the way mankind does things.

Next-generation 5G speeds will be 10 to 20 Gbps

ITU, the global spectrum-allocating organization, has reportedly agreed on speed criteria for 5G.

RELATED

What is 5G wireless? [And why it will change networking as we know it]

5G networks look to new frequencies to deliver gigabit speeds

Huawei 5G hits 3.6 gigabits per second in field test

VIDEO
What is NB-IoT?

The International Telecommunication Union (ITU) has defined 5G network speeds as being 20 Gigabits per second (Gbps), according to an article in the Korea Times.

However, an ITU spokesperson says it will be more like 10 Gbps with peak speeds at 20 Gbps, according to a separate Fierce Wireless report.

The ITU, a United Nations organization, has also come up with a name for the 5G standard – "IMT-2020."

Figure 5.4: "Next-generation 5G speeds will be 10 to 20 Gbps" news article

Figure 5.5: "Britain aims to join mobile broadband leaders with £35m '5G' research centre" news article

5.1.2 News Aggregator Dataset

News Aggregator Dataset is a dataset from the UCI Machine Learning Repository[5] [21]. The dataset contains 422937 news articles from 5-month interval: from March 10th to August 10th 2014. The original dataset does not contain content of news articles, hence content was crawled by web crawlers. Since articles come from 2014 majority of URLs had been outdated, content removed or moved to web archives. We managed to crawl 75% out of the total number of articles. Half of the retrieved articles were crawled from Wayback Machine Internet Archive[6].

Articles are from four categories: Business, Science & Technology, Entertainment, and Health. News articles in News Aggregator dataset had been collected from various online media sources (9311 unique sources). On average, each news story contains 58 news articles [22].

News Aggregator dataset does not contain news stories contents, except for alphanumeric ID of the cluster reflecting the news story. 82 news stories contain at least 200 articles (total 20041), 713 news stories contain 100 or more news articles (total 103957), 2202

[5] http://archive.ics.uci.edu/ml/datasets/News+Aggregator
[6] https://web.archive.org

news stories contain at least 50 articles (total 208505), 6129 news stories contain at least 10 articles (total 312755). News Aggregator dataset structure is shown in Table 5.3.

Table 5.3: News Aggregator Dataset structure

Field	Definition
ID	Numeric ID of a news article.
Title	Title of a news article.
URL	URL of a news article.
Publisher	Publisher of a news article.
Category	Category of a news article. Four categories in total: business (b), science and technology (t), entertainment (e) and health (m).
Story	The alphanumeric ID of a news story.
Hostname	The hostname where a news article had been published.
Timestamp	The publication date of a news article as an approximate Unix timestamp.

5.2 Experimental Metrics

To evaluate the performance of SVM and DSSM models we use following metrics.

1. Accuracy
2. Confusion matrix
3. Specificity
4. Precision
5. Recall
6. F1-score
7. Support

All these metrics are used in machine learning, as well as, in information retrieval, document classification, query classification and statistical analysis.

Confusion matrix

Confusion matrix, also known as confusion table or matching matrix, is used to visually

present classification results for each class in one table (matrix). Classification matrix contains information about correctly classified and misclassified instances. Each row in classification matrix represents instances of the predicted class, while each column represents instances of an actual class[7].

Classical confusion matrix for binary classification task is summarized in Table 5.4.

Table 5.4: Confusion Matrix for binary classification problem

	Actual class	
Predicted class	True Positive	False Positive
	False Negative	True Negative

True Positives (TP) indicate the number of correctly classified positive instances. True Negatives (TN) indicate the number of correctly classified negative instances. False Positives (FP), also known as type I error, indicate the number of negative instances which had been mistakenly classified as positive. For example, false fire alarm. False Negatives (FN), also known as type II error, indicate the number of positive instances which had been mistakenly classified as negative[7]. For example, fire alarm did not trigger when the fire started. For binary classification problem values of TP, TN, FP, and FN can be taken from the table, no other computations involved.

For multiclass problem the confusion matrix has a slightly different view, it is summarized in Table 5.5.

Table 5.5: Confusion matrix for mulitclass classification problem

		Actual class		
		Class 1	Class 2	Class n
Predicted class	Class 1	*val 1,1*	*val 1,2*	*val 1,n*
	Class 2	*val 2,1*	*val 2,2*	*val 2,n*
	Class n	*val n,1*	*val n,2*	*val n,n*

TP, TN, FP and FN metrics for each class can be calculated as followed:

1. TP is a diagonal element of the confusion matrix.

$$TP_{class_X} = CM[X_i, X_i]$$

[7] https://en.wikipedia.org/wiki/Confusion_matrix

2. TN is a sum of all values in the confusion matrix excluding row and column corresponding to the target class.

$$TN_{class_X} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j} - (row_{X_i} - column_{X_j} - TP_X)$$

3. FP is a sum of all values in a column of the confusion matrix corresponding to the target class excluding TP value.

$$TP_{class_X} = \sum column_X - TP_X$$

4. FN is a sum of all values in a raw of the confusion matrix corresponding to the target class excluding TP value.

$$TN_{class_X} = \sum raw_X - TP_X$$

5. The total number of class instances is a sum of a column corresponding to the target class.

$$N_{class_X} = \sum column_X$$

Accuracy

Accuracy is used to measure the proportion of correctly classified instances. Accuracy is calculated by using the formula:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

In some cases, accuracy is not reliable in measuring real performance of the classifier. Accuracy metric produces misleading results in case of imbalanced dataset [7]. For example, we have the dataset which consists of 10 entries of class A and 90 entries of class B. If all entries of class A are misclassified, the overall accuracy will be 90%. On practice, classifier shows 100% recognition rate for class B and 0% recognition rate for class A.

During our experiments, the training data is balanced and should contain the equal number of classes. The number of classes varies based on the selected number of positive and negative pairs.

Precision

Precision metric is also known as positive predictive value. Precision indicates the fraction of relevant instances out of total amount of retrieved instances[8]. In other words, precision

[8] https://en.wikipedia.org/wiki/Precision_and_recall

shows how useful the results are or the ability of the classifier to not label negative instances as positive[9]. Precision can be calculated by using the following formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall metric is also known as sensitivity, the true positive rate or probability of detection. Recall metric indicates the fraction of relevant instances out of total amount of relevant instances[8]. In other words, recall shows how complete the results are or the ability of the classifier to retrieve all positive elements[9]. Recall can be calculated by using the following formula:

$$Recall = \frac{TP}{TP + FN}$$

Specificity

Specificity metric is also known as true negative rate. Specificity measures the proportion of correctly identified actual negative instances[10]. In other words, specificity indicates how many selected negative elements are truly negative. Specificity can be calculated by using the following formula:

$$Specificity = \frac{TN}{TN + FP}$$

F1-score

F1-score is a metric which measures the harmonic average of precision and recall[11]. The minimum value of F1-score is 0 and the maximum value is 1. F1-score is calculated by using the formula:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2TP}{2TP + FP + FN}$$

Disadvantages of F1-score:

- F1-score is not a reliable metric for the imbalanced dataset[7].
- F1-score does not consider true negatives[11].

[9] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

[10] https://en.wikipedia.org/wiki/Sensitivity_and_specificity

[11] https://en.wikipedia.org/wiki/F1_score

5.3 Experimental Setup

We create the DSSM model in Keras[12] neural network API and run it on TensorFlow[13] backend - the open source machine learning framework. The model is trained on Tesla P100-PCI-E GPUs. All parameters used are summarized in Table 5.6.

Table 5.6: Model parameters

Parameter	Value	Description
num_articles	20	We select Wikipedia pages with at least 20 linked news articles.
k_folds	5	The number of folds used in K-fold cross-validation.
epochs	10	The number of epochs.
min_content_length	300	The minimum length of the content in characters. Shorter length articles will be discarded.
max_content_length	1000	The maximum length of the content in characters. Longer length articles will be trimmed, shorter ones will be sequence padded.
max_nb_words	20000	The maximum number of words.
embedding_dim	100	Embedding dimension from GloVe 6B.
validation_split	0.2	20% of data is used for validation.
J	3	For each news article, we generate 1 positive and J - 1 negative pairs.
optimizer	adadelta	Optimization algorithm used.
loss	categorical_crossentropy	Loss (cost) function used.

We test our model on the subset of data with at least 20 news articles linked to each Wikipedia page. We use 5-fold cross-validation (80% of data is reserved for training and 20% for testing), each fold is trained for 10 epochs. Minimum content length is 300 characters, maximum content length is 1000 characters, the maximum number of words is 20.000, embedding dimension is 100, validation split is 0.2 (20%). For each news article, we create one positive pair and two negative pairs (J = 3). We use "adadelta" optimizer and "categorical cross entropy" loss (cost) function. All the described parameters were chosen after performing a grid-search.

[12] https://keras.io/
[13] https://www.tensorflow.org/

Adadelta is an adaptive learning rate method for gradient descent introduced by Zeiler [23]. Adadelta was built on top of Adagrad with some improvements over the continual decay of learning rates and manual selection of global learning rate value. Different optimizers have one hyperparameter in common - learning rate. In practice, the learning rate is tuned manually. If learning rate is high, then performance will diverge. If learning rate is very small learning is becoming slaw with high error rate. Adadelta automatically computes the learning rate for each dimension by using first-order information, hence no manual tuning or grid search is required. Adadelta is not sensitive to hyperparameters, robust to large gradients and noise architectures and requires minimal computations over gradient descent.

Lately, Adam optimizer had been widely used, but for our model Adadelta reaches higher accuracy in less number of epochs without over-fitting.

We plot the accuracy and loss for each fold to visualize the training process. Accuracy and loss plots help to control training process, identify over-fitting and discard poor hyperparameter choices at early stages. An example of accuracy visualization is shown in Figure 5.6.

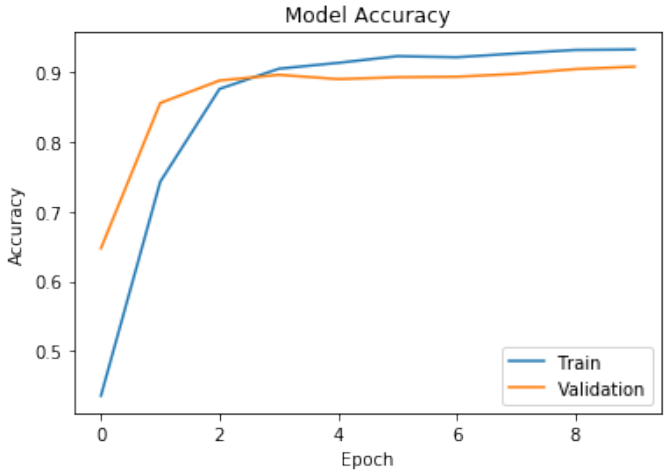


Figure 5.6: Accuracy visualization for the training process

If training loss keeps decreasing, but validation loss increases then model over-fits training data. The example of over-fitting visualized on loss plot is shown in Figure 5.7. After epoch number 15 model starts over-fitting. To achieve the best performance, the model should be trained for 15 epochs or early stopping mechanism should be implemented.

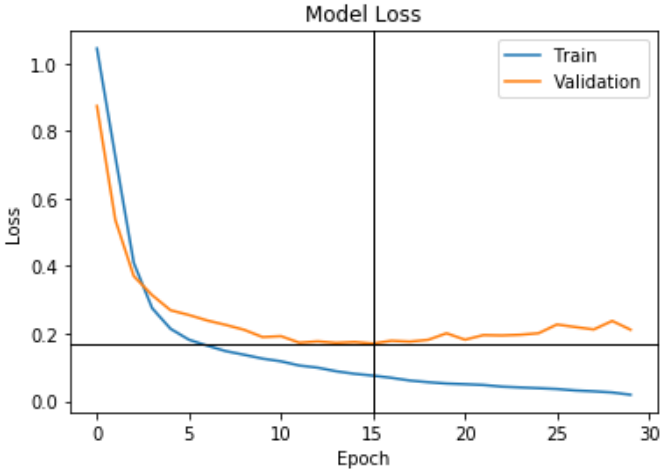


Figure 5.7: Over-fitting on loss plot

5.4 Experimental Results

5.4.1 Wiki Experiment

We test our model on the Wikipedia dataset. First, we select only news stories (events) which contain some minimal number of articles per event. Further, we go through all news articles and pair them up with news stories (Wikipedia pages, news events) to create training pairs.

5.4.2 Positive and negative pairs sampling

We consider multiple scenarios to sample positive and negative pairs.

Scenario 1
Wikipedia page content is used as a positive example. Negative examples consist of randomly selected Wikipedia pages the news article does not relate to.

Precision scores are presented in Table A.1, recall scores in Table A.2, specificity scores in Table A.3, F1-scores in Table A.4, and accuracy scores in Table A.5.

Scenario 2
From the Wikipedia dataset, we can extract sections of Wikipedia stories summarizing news articles. The Wikipedia page which summarizes a long-term event may contain different sub-topics. By using Wikipedia sections, as positive examples, we might improve the performance of the model. Negative examples are sampled randomly.

Not all news articles have Wikipedia section field filled, hence, we will use the section if the field is not empty. Otherwise, we use the whole content of Wikipedia story as a positive example.

Precision scores are presented in Table A.6, recall scores in Table A.7, specificity scores in Table A.8, F1-scores in Table A.9, and accuracy scores in Table A.10.

Scenario 3

Mitra et al. [24] suggest using similar documents as negative examples rather than sampling the negative documents uniformly. These documents would be classified by data assessors as not relevant. Positive examples are retrieved as described in Scenario 2.

We test two query setups for retrieving negative examples from Elasticsearch. Elasticsearch is an open-source, distributed, RESTful search and analytics engine[14]. Elasticsearch supports many types of search queries via provided full Query DSL (Domain Specific Language). Query DSL uses JSON to define the queries[15].

We first create Elasticsearch index for Wikipedia stories (pages). Elasticsearch index is a database which supports Elasticsearch queries. We do not need to create the index for news articles since we use Wikipedia stories to construct positive and negative pairs. Elasticsearch index for Wikipedia pages contains fields: index, title, and content. We use BM25 similarity model with default parameter values of $b = 0.75$ and $k_1 = 1.2$.

The logic is as follows:

1. Send search query to Elasticsearch Wikipedia stories index.
2. Retrieve top n search results (in the experiment we retrieve top 50).
3. Use last J (total number of positive and negative pairs) search results as negative examples.

We used two types of DSL search queries:

1. **Bool search query**
   Bool query matches documents based on the boolean combinations of other queries. Elasticsearch bool query uses boolean clauses: must, filter, should and must not[16]. See the query structure in 5.1.

[14] https://www.elastic.co/products/elasticsearch
[15] https://www.elastic.co/guide/en/elasticsearch/reference/6.2/query-dsl.html
[16] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html

```
query_json = {
   "query": {
      "bool": {
         "must_not": [
         {
            "match": {
               "title": story
            }
         }
         ],
         "should": [alternatives]
      }
   }
}
```

Listing 5.1: Bool query structure

Where each alternative is defined in 5.2.

```
{
   "multi_match" : {
      "query": "key_phrase",
      "fields": ["title", "content^2"]
   }
}
```

Listing 5.2: Key phrases multi-field queries

We ignore the Wikipedia story used as the positive example, hence, Elasticsearch must not return Wikipedia pages with the same title. We use news article title as a search query, but do it in a smart way. We extract the keywords and key phrases from news article title, add Wikipedia section name to key phrases list if it is available, and perform a multi-field search by values in the list. Multi-field search is performed by search in both: the title and the content fields with emphasis on the content field. All key search phrases are searched via should clause.

2. **Query string query**
   Query string query parses query string and analyses each part of the query independently[17]. We again search in title and content fields with an emphasis on the content field. We use the title of a news article as a query. See the query structure in 5.3.

[17] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html

```
query_json = {
   "query": {
      "query_string" : {
         "fields" : ["title", "content^2"],
         "query" : query
      }
   }
}
```

Listing 5.3: Query string query structure

Bool query precision scores are presented in Table A.11, recall scores in Table A.12, specificity scores in Table A.13, F1-scores in Table A.14, and accuracy scores in Table A.15.

Query string query precision scores are presented in Table A.16, recall scores in Table A.17, specificity scores in Table A.18, F1-scores in Table A.19, and accuracy scores in Table A.20.

Scenario 4

We consider the scenario when Elasticsearch is not always able to provide search results or return the necessary amount of documents. If negative pairs cannot be retrieved via Elasticsearch, we sample them randomly from the Wikipedia dataset.

Bool query precision scores are presented in Table A.21, recall scores in Table A.22, specificity scores in Table A.23, F1-scores in Table A.24, and accuracy scores in Table A.25.

Query string query precision scores are presented in Table A.26, recall scores in Table A.27, specificity scores in Table A.28, F1-scores in Table A.29, and accuracy scores in Table A.30.

All scenarios are summarized in Table 5.7.

Macro-averaged metric scores are shown in Table 5.8. Scenario 2 outperforms other scenarios. The DSSM was originally designed for short texts, mapping sections of the Wikipedia pages as positive examples gives a performance boost. We should keep in mind that Wikipedia sections are relatively long, several times longer than queries or document titles.

Elasticsearch query string query outperforms bool query by 1.8% in Scenario 3. In a hybrid mode (Scenario 4) for sampling negative examples, both queries perform equally, however, do gain 3.9% and 2.1% in accuracy correspondingly. Other metrics remain high scores as well.

We tried constructing the various number of negative pairs: from 2 to 15. When the number of negative pairs increases metric scores slightly drop. There is a trade-off between the number of negative pairs and training process speed.

Table 5.7: Summary of sampling scenarios

	Positive pairs	Negative pairs
Scenario 1	The full content of the Wikipedia story.	Randomly sampled from other Wikipedia stories.
Scenario 2	The section from the Wikipedia story if it is available, whole content otherwise.	Randomly sampled from other Wikipedia stories.
Scenario 3 bool	The section from the Wikipedia story if it is available, whole content otherwise.	Sampled from Elasticsearch via bool query.
Scenario 3 qstring	The section from the Wikipedia story if it is available, whole content otherwise.	Sampled from Elasticsearch via query string query.
Scenario 4 bool	The section from the Wikipedia story if it is available, whole content otherwise.	If possible, are sampled from Elasticsearch via bool query, random Wikipedia story otherwise.
Scenario 4 qstring	The section from the Wikipedia story if it is available, whole content otherwise.	If possible, are sampled from Elasticsearch via query string query, random Wikipedia story otherwise.

Table 5.8: Macro-averaged metric scores for sampling scenarios

	Precision	Recall	Specificity	F1-score	Accuracy
Scenario 1	0.897	0.888	0.944	0.889	0.888
Scenario 2	**0.977**	**0.975**	**0.988**	**0.975**	**0.975**
Scenario 3 bool	0.902	0.875	0.931	0.877	0.875
Scenario 3 qstring	0.912	0.893	0.946	0.895	0.893
Scenario 4 bool	0.927	0.914	0.956	0.915	0.914
Scenario 4 qstring	0.927	0.914	0.957	0.915	0.914

Confusion matrix for Fold 5 in Scenario 2 with total 963 test entries is shown in Table 5.9.

Table 5.9: Confusion matrix for Fold 5 Scenario 2

		Actual		
		Class 1	Class 2	Class 3
Predicted	Class 1	345	1	0
	Class 2	5	281	0
	Class 3	7	0	324

5.5 SVM baseline setup and results

We use SVM model as the baseline. SVM had proved to be suitable for document categorization task [2]. For SVM we reformulate the problem into binary. We group news articles with relevant and irrelevant Wikipedia stories the same way it is done in Scenario 2. For each text (news article/Wikipedia page content) we compute TF-IDF sparse vector representation. We aggregate (concatenate) TF-IDF vectors for each pair and append cosine similarity value computed between two vectors. Each training entry has binary label 0 or 1: 0 if we have the negative pair, 1 if the pair is positive.

Term frequency-inverse document frequency (TF-IDF) is widely used to describe the importance of a term in a document in a document collection. Term frequency reflects the importance of a term in a document. In other words, TF is the number of times that term t appears in document d divided by a number of documents which contain term t. Some words are more frequent than other. These words are called stop words. Stop words are less important than words that are not so frequent. Inverse Document Frequency (IDF) reflects the importance of the term in the collection of documents. IDF is calculated as a logarithm of a total number of documents in the collection divided by a number of documents which contain term t. By multiplying TF and IDF we receive combined TF-IDF weight as a feature [8].

We use SVC[18] implementation of SVM from the scikit-learn python machine learning library[19]. The SVC is initialized with "linear" kernel and "ovo" (one-against-one) decision function.

We use 3600 TF-IDF features for each content. Each training instance contains 3600 + 3600 + 1 features. SVM experimental results are shown in Table 5.10.

Table 5.10: Macro-averaged metric scores for SVM

	Precision	Recall	F1-score	Accuracy
SVM	0.558	0.648	0.554	0.648

As we can see, the best performing DSSM model for background linking task outperforms baseline SVM by 32.7% in accuracy, 41.9% in precision and 32.7% in recall. We should mention that all DSSM scenarios perform significantly better than the baseline model.

[18] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[19] http://scikit-learn.org/stable/index.html

5.6 Discussions

5.6.1 Regularization

Deep neural architectures tend to over-fit on smaller datasets. Regularization techniques are used to prevent a model from over-fitting [1]:

- L1 weight decay (least absolute deviations)
- L2 weight decay (least squared error)

$$w^* = argmin_w \sum_j (t(x_j) - \sum_i w_i h_i(x_j))^2 + \lambda \sum_{i=1}^{k} w_i^2$$

  Where λ is a regularization term.

- Early stopping
- Bagging (ensemble)
- Dropout
  Dropout prevents co-adaption of hidden units by removing hidden units stochastically.

When running the training process for 10 epochs the DSSM does not over-fit. After running the model for more than 10 epochs the model starts over-fitting (Wikipedia dataset is not very large in terms of modern datasets). We apply dropout with a probability of 0.5 and L2 regularization with $\lambda = 0.001$ when model tends to over-fit. For example, when running our best performing scenario for 30 epochs with regularization techniques, the model does not over-fit and shows the slight improvement in metrics, check Table 5.11.

Table 5.11: Macro-averaged metric scores Scenario 2 with regularization

	Precision	Recall	Specificity	F1-score	Accuracy
Scenario 2	0.977	0.975	0.988	0.975	0.975
Scenario 2 regularized	**0.98**	**0.978**	**0.989**	**0.978**	**0.978**

5.6.2 DSSM Applicability

The DSSM uses semantic similarity between two objects (query and document, news article and news event, etc.) to learn representations. Indeed, it is not suitable for

learning representations in non-semantic space. In other words, mapped objects should contain meaningful text.

Traditional IR approaches such as TF-IDF and BM25 only consider term counts (number of exact term matching) to rank documents by relevance (similarity) to the search query. They are not able to catch topical and typical notions of similarity.

Auto-encoders get compressed representations of text documents. The lengths of news articles vary, highly compressed representations fail to catch all the keywords and topics of news article content.

5.6.3 Challenges

One of the problems with news articles and events research field is the data itself. Other research topics have experimental datasets on which researchers can build benchmarks, perform experiments and compare results from other papers. For example, common datasets include MR (movie reviews), SST-1 (Stanford Sentiment Treebank with labeled movie reviews), SST-2 (movie reviews without neutral reviews), Subj (subjectivity dataset with subjective and objective sentences), TREC (TREC question dataset), CR (customer reviews) [19], Yelp reviews, IMDB reviews, Yahoo answers, Amazon reviews [25], etc.

There are no commonly accepted, balanced and publicly available news datasets. For example, News Aggregator dataset has news articles from 2014, does not contain contents of the news articles and news stories are not available. TREC challenge news datasets usually have data from one news media outlet (Washington Post), are not publicly available, does not include news events or news stories. In addition, Wikipedia does not have any dumps with Wikipedia pages and external sources linked to them. Not all external references are news sources. Many linked sources and external URLs are not active, had been moved to web archives or removed completely. Also, news contents are large and storing large datasets necessitates extra space and long processing times. Some datasets require hundreds of gigabytes of space on a hard drive.

Crawling news articles is a challenging procedure. Since no online media outlets provide the API for retrieving clean content, the title and content of the article need to be parsed from the HTML code of the page, which is noisy and inaccurate. News websites add commercials and advertisements to their pages which are retrieved along with HTML code. There are some external python libraries which help to automate this process. We used Newspaper3k[20] python library for article scraping as it gave better results than other alternatives.

[20] http://newspaper.readthedocs.io/en/latest/

Another challenge faced was that accessing the same domain multiple times in a short period of time leads to a ban or connection cut off. Hence, timeouts should be used to prevent such a behavior. Timeouts slow down the crawling process, making it less efficient. One of the solutions is to use multiple threads or processes. Many URLs are not active anymore and can be looked up only in web archives. For example, to crawl contents of News Aggregator dataset we used 29 parallel processes with distributed subsets of URLs to proceed, which auto-adjusted timeouts and checked web archives for missing content. Overall, it took 3 weeks of time to design scripts and crawl 75% of news contents. In addition, we found that some news online media have limited access and require the subscription. This means that content of these news articles is unavailable.

Chapter 6

Conclusion and Future Directions

6.1 Conclusions

In this research, we focused on solving background linking tasks for news articles by utilizing deep neural networks. We reviewed deep neural network structures and popular architectures. Throughout this work, the toolbox grew larger. We discussed types of learning, machine learning approaches to text categorization, neural and deep neural architectures. Since none of the neural networks work with raw text, we covered existing approaches to represent text with numerical vectors.

Background linking task implies the extraction of "similar" news articles for a given news article (or news story). To understand term "similar" we discussed similarity metrics and notions of similarity. We noticed some similarities between the background news articles linking task and information retrieval problem. Inspired by IR approaches we adjusted Deep Semantic Similarity Model to enhance the background linking task. We used GloVe 6B pre-trained word embeddings which catch topical and typical similarities.

We designed a convolutional neural network with multiple filters, convolutional and max-pooling layers to retrieve news article representations. Word embeddings were input to the convolutional network. Convolutional layers extracted local features, max-pooling layers extracted global features such as key topics and keywords. The semantic layer is the output of the CNN fully connected layer and contains a vector representation of the news article.

DSSM architecture implies usage of positive and negative pairs. Source article is denoted as X, positive (similar, within the same news story) articles as Y^+ and negative (not relevant, from other news stories) as Y_-. Cosine similarity is used to compute similarity between the articles in semantic space. Similarity values are then passed to the softmax

posterior probability module. The goal of the training was to increase the distance between positive and negative pairs.

We tested our adjusted DSSM model on the Wikipedia dataset and compared it to Support Vector Machine baseline. SVM uses TF-IDF from articles as features. In training process, we tested different scenarios to populate positive and negative training pairs. In all scenarios, DSSM model proved to be suitable for retrieving background news articles. The best-case scenario shows 32.7% gain in accuracy, 41.9% in precision, 32.7% in recall in comparison to SVM baseline.

The DSSM model can also be used to find the nearest category or event, and for automated linking news articles to Wikipedia pages. We discovered that deep neural networks are suitable architectures which identify complicated patterns and approximate any function.

6.2 Future Work

In Wikipedia dataset, we used linked Wikipedia pages (stories) as positive examples. A Wikipedia story is a summary of the whole event, short and precise, with a guaranteed level of quality. News Aggregator dataset does not have such event summaries. The goal of the News Aggregator dataset is to cluster news articles related to distinct stories [22]. News Aggregator dataset has no ground truth stories content, except alphanumeric string representing news clusters (stories). Hence, other approaches should be used to apply DSSM to News Aggregator dataset. The first approach; use random news articles from the same cluster to construct a positive pair. The second approach; use one news article within the same cluster as a positive example for all other articles within the same cluster. Negative pairs should be either sampled randomly or retrieved via Elasticsearch.

Text Retrieval Conference (TREC) conducted by the National Institute of Standards and Technology (NIST) annually announces research contests called tracks[1]. One of the challenges relevant to the research topic is News Track[2]. News Track has two tasks: background linking and entity ranking. Background linking task requires the retrieval of relevant news articles for a given news story. One of the directions for future work is to adapt and test DSSM model to TREC news dataset. First, it requires some manipulations with the dataset. We started, but since guidelines had been released late (11-05-2018[3]) it cannot be performed in this research.

[1] https://trec.nist.gov/

[2] http://trec-news.org/

[3] https://docs.google.com/document/d/e/2PACX-1vSJvm30NV4aT4fRcf6x-J-AjvZqaWEw8DsjgXP1v3NlcWZZEtxZ9SwmuB-sQvcc_G7ER-BcUKJQoZHn/pub

Another direction of future work is to apply the DSSM to automatically link news articles to Wikipedia pages. All news articles are linked to Wikipedia pages manually, hence automating this process can be of significant importance.

In this research, we had been learning news article representation. In practice, it is also useful to learn the representation of a news event by designing auto-encoder to learn the representation of news events from the sequence of news articles on a timeline and use it to retrieve similar news events. The DSSM architecture can be used not only to retrieve similar news articles within news story (event) but also to find similar events.

First Law of Thermodynamics states that nothing comes from nothing and nothing vanishes without any marks. In real life, however, many events are consequences resulting from another related event. Here, we talk about the chain of events. When you start reading articles on the internet, the question arises: is it a one-time, stand-alone event or one in the sequence of other events? For example, articles about burned animals in California 2017 is not a stand-alone event, this event is a part of a long-running event about California wildfires. Some chains of events have cyclic character, like FIFA World Cup, Winter and Summer Olympics which take place once in four years. Our goal was to link similar news events and stories in an expedited manner for the reader. While we have shown some success in our endeavor, more work needs to be done to solidify the use of our adjusted Deep Semantic Similarity Model.

List of Figures

List of Tables

Appendix A

Detailed metrics tables

In this appendix we list detailed tables with metrics for sampling scenarios. Each table shows micro-average statistics for each fold and macro-average value. For every scenario we show precision, recall, specificity, F1-score and accuracy metrics results.

1. **Scenario 1**
   Tables A.1, A.2, A.3, A.4 and A.5.
2. **Scenario 2**
   Tables A.6, A.7, A.8, A.9 and A.10.
3. **Scenario 3**
   - *Elasticsearch bool query*
     Tables A.11, A.12, A.13, A.14 and A.15.
   - *Elasticsearch query string query*
     Tables A.16, A.17, A.18, A.19 and A.20.
4. **Scenario 4**
   - *Elasticsearch bool query*
     Tables A.21, A.22, A.23, A.24 and A.25.
   - *Elasticsearch query string query*
     Tables A.26, A.27, A.28, A.29 and A.30.

Table A.1: Scenario 1 test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.806	0.832	0.817	0.799	0.793
2	0.91	0.962	0.984	0.902	0.963
3	0.95	0.882	0.92	0.973	0.968
Micro-average	0.886	0.892	0.905	0.892	0.911
Macro-average	0.897				

Table A.2: Scenario 1 test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.955	0.925	0.968	0.943	0.976
2	0.842	0.822	0.851	0.868	0.879
3	0.83	0.912	0.864	0.833	0.847
Micro-average	0.878	0.886	0.895	0.882	0.899
Macro-average	0.888				

Table A.3: Scenario 1 test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.874	0.904	0.887	0.878	0.883
2	0.961	0.983	0.994	0.955	0.982
3	0.979	0.941	0.961	0.988	0.986
Micro-average	0.938	0.943	0.947	0.940	0.950
Macro-average	0.944				

Table A.4: Scenario 1 test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.874	0.876	0.886	0.865	0.875
2	0.875	0.886	0.912	0.885	0.919
3	0.886	0.896	0.891	0.898	0.904
Micro-average	0.878	0.886	0.896	0.883	0.9
Macro-average	0.889				

Table A.5: Scenario 1 test Accuracy

	Fold1	Fold2	Fold3	Fold4	Fold5
	0.878	0.886	0.895	0.882	0.899
Average	0.888				

Table A.6: Scenario 2 test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.947	0.926	0.946	0.894	0.966
2	1	1	0.994	1	0.996
3	0.997	1	0.996	1	1
Micro-average	0.979	0.975	0.978	0.966	0.987
Macro-average	0.977				

Table A.7: Scenario 2 test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.997	1	0.997	1	0.997
2	0.976	0.945	0.976	0.913	0.983
3	0.958	0.972	0.956	0.97	0.979
Micro-average	0.978	0.973	0.977	0.962	0.987
Macro-average	0.975				

Table A.8: Scenario 2 test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.967	0.959	0.97	0.943	0.981
2	1	1	0.997	1	0.999
3	0.998	1	0.999	1	1
Micro-average	0.988	0.986	0.989	0.981	0.993
Macro-average	0.988				

Table A.9: Scenario 2 test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.971	0.962	0.971	0.944	0.982
2	0.988	0.972	0.985	0.955	0.989
3	0.977	0.986	0.976	0.985	0.989
Micro-average	0.978	0.973	0.977	0.962	0.987
Macro-average	0.975				

Table A.10: Scenario 2 test Accuracy

	Fold1	Fold2	Fold3	Fold4	Fold5
	0.978	0.973	0.977	0.962	0.987
Average	0.975				

Table A.11: Scenario 3 bool query test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.671	0.776	0.762	0.725	0.796
2	0.987	0.981	0.991	0.985	0.985
3	0.961	0.992	0.98	0.994	0.974
Micro-average	0.867	0.915	0.908	0.901	0.919
Macro-average	0.902				

Table A.12: Scenario 3 bool query test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.984	0.983	0.976	0.99	0.971
2	0.657	0.849	0.794	0.781	0.864
3	0.793	0.851	0.879	0.834	0.884
Micro-average	0.817	0.896	0.887	0.869	0.906
Macro-average	0.875				

Table A.13: Scenario 3 bool query test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.738	0.854	0.84	0.811	0.878
2	0.996	0.992	0..997	0.994	0.993
3	0.984	0.997	0.991	0.997	0.988
Micro-average	0.906	0.948	0.916	0.934	0.953
Macro-average	0.931				

Table A.14: Scenario 3 bool query test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.798	0.867	0.856	0.837	0.875
2	0.789	0.91	0.882	0.871	0.92
3	0.869	0.916	0.927	0.907	0.927
Micro-average	0.818	0.898	0.889	0.872	0.907
Macro-average	0.877				

Table A.15: Scenario 3 bool query test Accuracy

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
	0.817	0.896	0.887	0.869	0.906
Average	0.875				

Table A.16: Scenario 3 query string query test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.798	0.746	0.792	0.799	0.749
2	0.984	0.978	0.978	0.977	0.99
3	0.972	0.991	0.968	0.992	0.976
Micro-average	0.915	0.904	0.91	0.923	0.906
Macro-average	0.912				

Table A.17: Scenario 3 query string query test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.982	0.988	0.975	0.985	0.979
2	0.866	0.839	0.815	0.875	0.835
3	0.844	0.801	0.888	0.865	0.837
Micro-average	0.9	0.877	0.896	0.908	0.883
Macro-average	0.893				

Table A.18: Scenario 3 query string query test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.865	0.827	0.865	0.876	0.84
2	0.993	0.991	0.992	0.99	0.996
3	0.988	0.997	0.984	0.997	0.99
Micro-average	0.949	0.938	0.947	0.954	0.942
Macro-average	0.946				

Table A.19: Scenario 3 query string query test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.881	0.85	0.874	0.882	0.849
2	0.921	0.903	0.889	0.923	0.906
3	0.904	0.886	0.926	0.924	0.901
Micro-average	0.901	0.88	0.897	0.91	0.886
Macro-average	0.895				

Table A.20: Scenario 3 query string query test Accuracy

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
	0.9	0.877	0.896	0.908	0.883
Average	0.893				

Table A.21: Scenario 4 bool query test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.8	0.773	0.799	0.85	0.835
2	0.998	0.996	0.985	0.983	0.994
3	0.979	0.996	0.989	0.973	0.984
Micro-average	0.922	0.92	0.922	0.935	0.938
Macro-average	0.927				

Table A.22: Scenario 4 bool query test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.994	0.995	0.989	0.993	0.993
2	0.856	0.861	0.851	0.874	0.882
3	0.86	0.836	0.875	0.914	0.913
Micro-average	0.906	0.898	0.907	0.928	0.929
Macro-average	0.914				

Table A.23: Scenario 4 bool query test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.865	0.849	0.869	0.911	0.904
2	0.999	0.998	0.994	0.992	0.997
3	0.991	0.998	0.995	0.987	0.992
Micro-average	0.952	0.948	0.953	0.963	0.964
Macro-average	0.956				

Table A.24: Scenario 4 bool query test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.886	0.87	0.884	0.916	0.907
2	0.921	0.924	0.913	0.925	0.935
3	0.916	0.909	0.928	0.943	0.947
Micro-average	0.907	0.901	0.908	0.928	0.93
Macro-average	0.915				

Table A.25: Scenario 4 bool query test Accuracy

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
	0.906	0.898	0.907	0.928	0.929
Average	0.914				

Table A.26: Scenario 4 query string query test Precision

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.806	0.793	0.824	0.838	0.803
2	0.986	0.986	0.968	0.982	0.99
3	0.99	0.982	0.993	0.989	0.987
Micro-average	0.924	0.919	0.927	0.936	0.928
Macro-average	0.927				

Table A.27: Scenario 4 query string query test Recall

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.994	0.987	0.99	0.987	0.985
2	0.859	0.853	0.884	0.905	0.846
3	0.867	0.869	0.87	0.889	0.914
Micro-average	0.909	0.903	0.916	0.927	0.914
Macro-average	0.914				

Table A.28: Scenario 4 query string query test Specificity

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.87	0.867	0.889	0.904	0.882
2	0.994	0.994	0.987	0.992	0.996
3	0.996	0.993	0.997	0.995	0.994
Micro-average	0.953	0.951	0.958	0.964	0.957
Macro-average	0.957				

Table A.29: Scenario 4 query string query test F1-score

Class	Fold1	Fold2	Fold3	Fold4	Fold5
1	0.89	0.879	0.9	0.906	0.885
2	0.918	0.915	0.924	0.942	0.912
3	0.924	0.922	0.927	0.936	0.949
Micro-average	0.91	0.905	0.917	0.928	0.916
Macro-average	0.915				

Table A.30: Scenario 4 query string query test Accuracy

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
	0.909	0.903	0.916	0.927	0.914
Average	0.914				

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nedellec and Céline Rouveirol, editors, *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer, 1998. ISBN 3-540-64417-2. doi: 10.1007/BFb0026683. URL https://doi.org/10.1007/BFb0026683.

[3] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[4] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *CoRR*, abs/1705.01509, 2017. URL http://arxiv.org/abs/1705.01509.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL http://arxiv.org/abs/1406.2661.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. ISBN 978-1-937284-96-1. URL http://aclweb.org/anthology/D/D14/D14-1162.pdf.

[7] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. ISBN 0-321-32136-7.

[8] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines - Information Retrieval in Practice*. Pearson Education, 2009. ISBN 978-0-13-136489-9. URL http://www.search-engines-book.com/.

[9] *Learning Deep Structured Semantic Models for Web Search using Clickthrough Data*, October 2013. ACM International Conference on Information and Knowledge Management (CIKM). URL https://www.microsoft.com/en-us/research/publication/learning-deep-structured-semantic-models-for-web-search-using-clickthrough-data/.

[10] *Learning Semantic Representations Using Convolutional Neural Networks for Web Search*, April 2014. WWW 2014. URL https://www.microsoft.com/en-us/research/publication/learning-semantic-representations-using-convolutional-neural-networks-for-web-search/.

[11] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. CIKM, November 2014. URL https://www.microsoft.com/en-us/research/publication/a-latent-semantic-model-with-convolutional-pooling-structure-for-information-retrieval/.

[12] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24:694 – 707, January 2016. URL https://www.microsoft.com/en-us/research/publication/deep-sentence-embedding-using-long-short-term-memory-networks-analysis-application-information-retrieval/.

[13] Scott Wen-tau Yih, Xiaodong He, and Chris Meek. Semantic parsing for single-relation question answering. Association for Computational Linguistics, June 2014. URL https://www.microsoft.com/en-us/research/publication/semantic-parsing-for-single-relation-question-answering/.

[14] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. ACL âĂŞ Association for Computational Linguistics, July 2015. URL https://www.microsoft.com/en-us/research/publication/semantic-parsing-via-staged-query-graph-generation-question-answering-with-knowledge-base/.

[15] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. Modeling interestingness with deep neural networks. Technical report, October 2014. URL https://www.microsoft.com/en-us/research/publication/modeling-interestingness-with-deep-neural-networks/.

[16] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, Larry Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. IEEE - Institute of Electrical and Electronics Engineers, June 2015. URL https://www.microsoft.com/en-us/research/publication/from-captions-to-visual-concepts-and-back/.

[17] Jianfeng Gao. Introduction to deep learning for natural language processing (tutorial at deeplearning2017 summer school in bilbao). Technical report, July 2017. URL https://www.microsoft.com/en-us/research/publication/introduction-deep-learning-natural-language-processing-tutorial-deeplearning2017-summer-school-bilbao-2/.

[18] Jianfeng Gao, Xiaodong He, and Li Deng. Deep learning for web search and natural language processing. Technical report, January 2015. URL https://www.microsoft.com/en-us/research/publication/deep-learning-for-web-search-and-natural-language-processing/.

[19] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

[20] Vinay Setty, Abhijit Anand, Arunav Mishra, and Avishek Anand. Modeling event importance for ranking daily news events. In Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang, editors, *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 231–240. ACM, 2017. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661. URL http://dl.acm.org/citation.cfm?id=3018728.

[21] M. Lichman. Uci machine learning repository. irvine, ca: University of california, school of information and computer science., 2013. URL http://archive.ics.uci.edu/ml.

[22] Fabio Gasparetti. Modeling user interests from web browsing activities. *Data Min. Knowl. Discov.*, 31(2):502–547, 2017. doi: 10.1007/s10618-016-0482-x. URL https://doi.org/10.1007/s10618-016-0482-x.

[23] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

[24] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. *CoRR*, abs/1610.08136, 2016. URL http://arxiv.org/abs/1610.08136.

[25] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In

Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489. The Association for Computational Linguistics, 2016. ISBN 978-1-941643-91-4. URL http://aclweb.org/anthology/N/N16/N16-1174.pdf.