Eivind Havikbotn

Deep Neural Models for Question-Answering Retrieval

Specialization project,
Fall 2016
TDT4501

Under the guidance of Massimiliano Ruocco

Data and Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

Abstract

In this project we investigate different Deep Learning approaches for question-answering retrieval, primarily based on research from Microsoft and the IBM Watson group. After a somewhat troublesome development period we ended up with five different architectures. We tested these models on four different datasets, the InsuranceQA dataset created by [13], and post-reply pairs retrieved from the Facebook pages of major telecommunication companies in Norway. Two of the models were built on a hash-function based word representation with convolutional filtering, and the remaining three were based on Word2Vec[22] embeddings combined with feed forward, convolutional and recurrent layers. We found that the best performing model was the architecture featuring Word2Vec embeddings for word representation, combined with a LSTM layer for word sequence modeling topped with a convolution layer for semantic extraction. We evaluated the models by MMR, Top1, Top5 and Top20 scores, and we performed a simple semantic evaluation by sampling post-reply pairs with predicted and true answers. In addition we used t-SNE [21] to perform a visualization of the model generated embedding space.

The source code for the project are available on the following url: https://github.com/eivhav/DeepQA

Contents

Chapter 1

Introduction

In this project we explore different approaches for building a non-factoid question and answering system. Due to the latest advancements in the field of Deep Learning based Natural Language Processing combined with the emergence of large amounts of conversational data publicly available at social media sites such as Twitter and Facebook, many organizations are seeking technology that can help them better interact with their customer by natural language, in addition to better understand their customers perception of the organization in the public sphere.

1.1 Background

Understanding natural language has been a devoted challenge since the dawn of computing and true language understanding is often described as a AI-complete problem, that is a problem that can not be solved without creating human level artificial intelligence. However many solutions can give near human level performance in more narrow, closed domains. In the early days of AI, NLP system where typical rule based systems, either by the designers limited knowledge about the domain, or by methods based on decision tree generated part-of-speech tags. In other words systems where information are retrieved by looking at known grammatical relationships, or parse trees. These system has often proved not to scale well once their faced with questions outside their preconceived domain and vocabulary.

In recent years many researchers has turned their focus towards neural net implementations for the

NLP task. Much due to research such as Thomas Miklov's popular Word2Vec [22] and Standford's GLove [3], which enables large vocabularies to be represented with a dense semantically relevant representation, combined with better understanding of recurrent neural networks that are added to adapt memory when reading sentences from start to end. Recent advancements in hardware has also played i major role, in particular the fact that researchers now can train large models with millions of parameters over dataset with billions of words on parallel implementations on low cost GPU's, which has enabled quick model feedback and the ability to generalizes models over larger and larger datasets.

It has however taken some time for consumer based applications to reach the marked, that is applications that can accomplish anything more than trivial commends and queries. Having a natural conversation with a bot seems a bit far away, but it's not completely unrealistic to think that dialog systems will become the dominant way of customer-service interaction in 5-10 years.

The two most common approaches for building dialog systems can be described as:

1. **Retrieval based Models** - The system is build with the goal of matching a message(query) with the most relevant reply in a fixed corpus of replies. Not feasible in practice if the system is to operate in a open domain, that is the user can ask the bot anything, combined with taking the conversation anywhere. However if the domain is defined, as in the domain for a customer-service bot, modeling most responses could be achievable.
2. **Generative Model** - Instead of utilizing predefined responses, the system can be built to generate the responses by statistical machine translation models, where the model generates a response word-sequence, given a sequence of input words [25]. These models are on the other significantly harder to train, and are more likely to produce grammatically incorrect sentences as output. Therefor in this project we will exclusively work on the retrieval based approach.

1.2 Objectives of the work

The goal of the project is to explore different approaches for implementing neural network models for semantic representation of questions and their answers. By training the models with different variants of the backpropagation algorithm, where the resulting semantic representation of questions and answers are close, the model can be used for answer retrieval/ranking. In addition we hope to be able to visualize the semantic space to see clustering tendency between related questions.

We focus primarily on two sets of data:

1. The InsuranceQA dataset [13], which contains 12000 insurance related questions and answers from the site InsuranceLibary.com
2. Data retrieved from the social media platform Facebook, where we obtain post/reply pairs from all major telecom companies in Norway

The InsuranceQA dataset is a commonly used dataset for answer retrieval research, including in [27], and it will give us a baseline to compare how well our experimental models performs, especially during topology selection and hyperparamter tuning. The social media dataset is on the other hand more complex in terms of grammatical incorrectness, different dialects, and references to entities(products, locations etc). In addition, it contains a high amount of words with low frequency, which might prove challenging. We will explore how well the different architectures will generalize under these conditions.

1.3 Report overview

In chapter 2 we will give an overview over the basic theory behind different neural network components needed to understand the project. This includes feed-forward networks, convolutional layers and recurrent networks.

In chapter 3 we will present the state-of-the-art for neural network implementations for text processing, and the most prominent approaches to solve our question-to-answer matching task.

In chapter 4 we will take a look at our datasets, our experimental models and the challenges faced during implementation and training.

In Chapter 5 we will present the experimental setup and the results. We will evaluate the different models by common metrics such as Mean Reciprocal Rank and TopK scores, and we will perform an objective evaluation by looking at the top-ranked and true answers. At the end of the chapter we will present a visualization for the embedding space for subset of the questions.

In the last chapter we will revise our results, make our conclusions and discuss further work needed for the system to work better.

Chapter 2

Background Theory

2.1 Neural networks basics

Although neural networks(often referenced under the popular term Deep Learning) has recently become a hot topic in the computer science community, the idea can be traced back to as early as the 1950's. The basic concept is to connect a series of activation nodes together by trainable weights, so the model can learn a complex input to output function. The concept is inspired by the human brain where biological neurons are connected by synapses, and signals are propagated to produce brain activity such as thought, actions and learning.

A simple example is the feed-forward network. A set of input nodes are fully connected to stack of layers, where each layer l_n is fully connected to layer l_{n+1}. In each connection a unique weight is stored, resulting in that each node receives a total incoming value of sum(l_{n-1} * w_i). Upon input, each node determines their output value by a activation function(usually Relu, sigmoid, or the hyperbolic tangent function), and the output value is then propagated to the nodes in the next layer. In mathematical terms the input to a feed forward layer can be described by the matrix multiplication Y = W x X, where W is the |X x Y| weight matrix. The last layer of the network can be seen as the models output, and by training the model to match the output to a correct representation, usually by defining a loss function based upon soft-max or cross entropy, the network can learn the desired input to output mapping after performing weight readjustments upon each forward propagation.

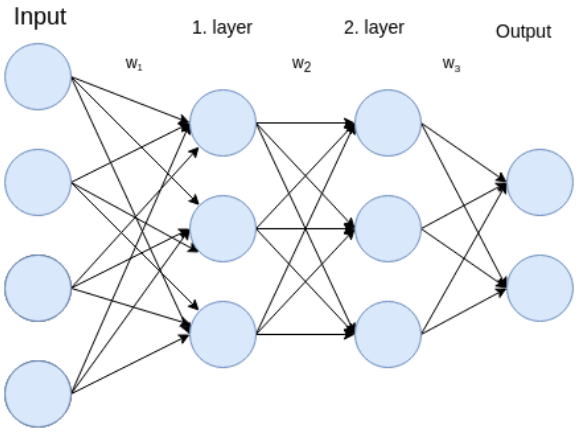

Figure 2.1: Simple feed forward network with 3 layers

2.2 Convolutional neural nets

However in many applications learning a connection between all input nodes to all output nodes may not be desirable. Instead we might be interested in local patterns. For example for an image we might be interested in the color difference of a few pixels, and for text classification the ordering of two words in a certain word-embedding dimension.

For these task we can employ a Convolutional neural net layer, which can simply be described as a sliding feature extractor applied over the input data. Given a kernel of size k and a corresponding kernel weight matrix W, the component slides over the input data, multiplying the input for the given window at each time step by the kernel weight matrix. As a result we retrieve a kernel-wise summary of the input for each local area by size k. By then applying a pooling operation over all local areas, we can detect edges and other important features in the input data.

Convolutional neural nets has been widely adopted for image recognition, much due to it's capabilities for capturing objects and patterns of different size regardless of their data-wise position. Google's Inception architecture and Microsoft's ResNet [26] are example of this, where several convolution and max-poling layers are stacked on top of each other, each layer/block with the task of representing the image at a higher level of abstraction. Although many researchers focus on recurrent neural nets for natural language processing, as we will see in this report, convolutional layers can also be useful for text modeling.

2.3 Recurrent neural nets

Recurrent neural nets is another architect-component, that has proved itself particularly powerful on sequence-oriented data. The main concept is to reconnect some part of the output(from a single node or entire layer), and then feed the signal back into the module when computing the next data input. In this manner the model can incorporate memory, since the output depends not only on the current input but all previous inputs. This often works well for types of data where there is a latent relationship between the input samples, by example in language modeling where a word at time t+1 has a relationship to the previous word at time t.

However, when RNN's are used over long input sequences, early propagated information can often become lost. The reason is that more and more data are propagated through the same weights, which makes it hard to learn long term dependencies. This has led to the development of optimized variants such as the Long-Short-term-Memory network which was first introduced in [17]. The basic idea is that each cell contains a state which is updated by using several internal trainable weights. Each cell receives the input together with the state from the last time-step, and from here the cell computes how much of the last-state information should be "forgotten", and how much to forward.

Chapter 3

State Of The Art

3.1 Deep structured semantic model

We begin our research by looking into an architecture for language modeling proposed by Huang et al in [18]. In short, the authors utilizes a feed-forward neural network for query to document matching, trained on click-through data from a web-search engine.

The proposed model inputs a word-indexed vector and converts it to a multi-hot representation with the following word-hashing technique. The idea is that a word can be represented by it's 3-letter permutations, by example the word "good" can be represented by the tri-letters 'go', 'goo', 'ood' and 'od'. By this method, the vector representation for each word can be reduced from the real vocabulary size (which can be anywhere from 100k to several millions), to an upper bound size of $(token - size)^3$. This can be further reduced to only contain tri-letters actually observed in the training corpus, since a portion of all potential tri-grams are not part of real words. Naturally there will be some collision by using this technique, but the authors estimate the collision rate to be the negligible.

The word hashing vector is then connected to a 3 layer non-linear projection network, where the last layer represents the embedding vector for that sentence/document. During training, the semantic vector for a query is optimized to be similar to the semantic vector of the relevant document, and at the same time dissimilar to non relevant documents. One important feature of the DSSM model is that all tri-letter representations for a document are merged into the same vector, which reduces

the representation from contextually rich input to a bag-of-tri-letters count over all sentences. The effect is naturally severe loss of semantic information contained in the word-wise structure, since word positions are ignored.

3.2 Convolutional DSSM

In [23] Shen et al tries to overcome to contextual loss problem imposed by the DSSM model, by representing each word as a tri-gram constructed by the neighbouring words. The idea is that words often have a different meaning depending on their surrounding words, by example the word office have a different meaning when together with Microsoft and excel, in contrast with building and corporate.

The approach is to reuse the word hashing method proposed by [18], but instead represent each word by concatenating the hashing vectors for w(j-1), w(j) and w(j+1) for each word w(j). Then, instead of merging the hashing vectors as in [18], a 1-dimensional convolutional operation is performed over the word concatenations that captures both the tri-letter word-representation and the contextual position of the words. The output is a matrix with dimensions K x W, where K is the number of filters used in the convolution operation and W is number of words in the sentence. K is in the paper set to be 300.

Since the amount of words can vary from sentence to sentence, a max operation is applied over each dimension in K, where the most dominant tri-gram at that dimension contributes with it's value. The output is now fixed to K dimensions regardless of the sentence length, and in order to learn a more complex embedding value, the max-pooling layer can afterwards be connected to a single feed-forward layer, where the output represents the final embedding from the sentence/document.

In the above mentioned paper the model is trained successfully on query title pairs outperforming other techniques such as Okapi BM25 and Latent Dirichlet allocation, in addition to the non convolutional DSSM model mentioned above. The same architecture has also been successfully applied in [24] where Shen et al proves that the model also works with query to document-body samples.

In [16] the authors reuses the architecture mentioned above to model interestningness between a web-search query and document, defined as either automatic highlighting or contextual entity match. By automatic highlighting they are interested in discovering the k most interesting keywords for the target document. Contextual entity search entails finding entities in the document that the

user has of interest given the search, and discarding entities that are not semantically relevant to the query. After training their model on a large click-trough data-set, their resulting model outperforms several of non-neural methods such as BM25 and BLTM [15]

3.3 Word embeddings, Word2Vec

A different approach for word representation is the embedding technique proposed by Thomas Mikolov in [22]. His approach was to train a feed-forward network to learn a mapping between a word and it's surrounding words in a fixed window size. The model can either be trained by inputting the surrounding words, and then optimize the model to predict the word in question(CBOW architecture), or by using the target word as input and attempt to predict the surrounding words (Skip-gram model). All words share the same weight matrix, but since the input are one-hot vectors, only the weights that corresponds to the words used in the window will be updated during training.

After the model has been trained, the resulting weights can be used as a smaller and denser representation for each word. In addition, each embedding will now contain additional semantic and syntactic information, where similar words are close in the embedding space and real world relations can be found by performing vector manipulation on the embedded entities. For instance, by calculating the vector(queen) - vector(woman) + vector(man), the resulting vector will most likely have the highest similarity with the vector for king.

In [22], Mikolov runs experiments on both the Skip-gram and CBOW model, in addition to previously proposed models with higher complexity. The models are trained on dataset ranging from hundreds of millions to billions of words. The authors points out that in order to achieve greater accuracy, one would be required to both increase the size of the word embeddings and the training data size. Training small embeddings on larger datasets often yields little accuracy gain. In addition the skip gram model outperforms the CBOW model significantly in terms of semantic accuracy, but training complexity is roughly three times higher.

As we will see later, after training the word embedding's for our corpus, a simple semantic question-answering model can be build by applying a max pooling operation at each dimension, gathering the most prominent features for each axis. This is quite similar for the cDSSM model, except that the text is represented as bag-of-words instead of bag-of-trigrams. As we will see in the result section, this simple model works surprisingly well, and has also been adopted for visual to text embeddings[14].

3.4 Convolutional based Models

In order to preserve longer contextual information, the bag-of-words or bag-of-trigrams models discussed until now will not be sufficient. We need a method that can capture the latent word dependencies for words further apart. One possible solution is to use a convolutional layer as described in chapter 2.

Once the sentence has been represented as a word-embedding matrix(typical word2vec or Glove), convolutional filters can be added on the sentence matrix where the sliding feature extractor(kernel), summarizes the pattern for the sequence of words(embedding values) at each specific dimension. In [19] the author Yoon Kim, applies this technique on several datasets for sentiment analysis and Q/A retrieval, by using multiple filters of different sizes. In that manner, each filter with size k, slides over k words per time-step, thereby extracting the contextual information for k words at a time. By afterwards max-pooling the value for each filter, the most prominent value of the different time-steps are preserved for each filter. Kim's model can be seen in figure (3.1).

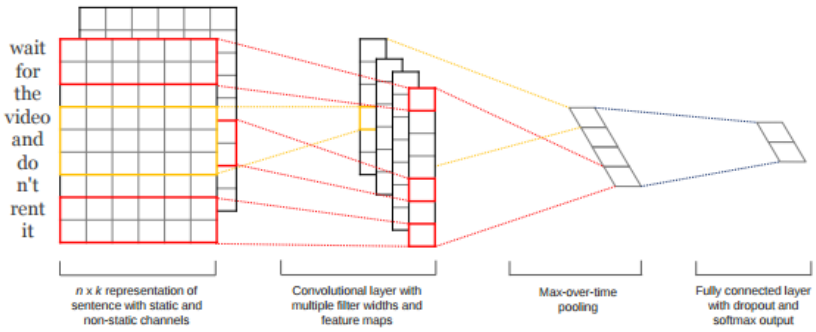

Figure 3.1: Architecture and figure from [19].

In [13], the authors follows the same methodology as in [19] by using convolutional layers of different filter sizes and widths, to a series of models for the question-answering task. They experiment with different architectures where each model differs in topology selection by placement of hidden layers and weight sharing between the question and answer parts, in addition to filter widths and number of filters. Their best model includes a feed-forward layer of size 200 connected from the word embeddings to a convolution layer with filter size 4000. In addition they found that weight sharing between the question and answer parts gave better results than learning the weights separately.

3.5 Bidirectional-LSTM models

Since sentence can be seen as a sequence of inputs, utilizing recurrent neural networks seems like an obvious choice for many NLP tasks. Especially if we employ a LSTM[17] or GRU network, where the vanishing/exploding gradients problem are not a big concern. However, when using such components for text classification, we still can face the problem with loss of activations for early words long sentences. One solution is to use a Bidirectional LSTM, which entails deploying two separate LSTM's where one of them reads the sentence as normal and the other reads the word sequence in reverse order. The resulting output is then the concatenation of the outputs from both LSTM modules.

In [27] Tan et al build on top of the work done by [13] by first building an BiLSTM model, where word embeddings are trained by Word2vec and used as input to the BiLSTM, and thereafter utilizing a max-pooling layer where for each dimension the most prominent word(time-step) are retrieved. The output is an embedding vector for that input sentence, which are optimized to match between question and answer. In addition they extend the model by replacing the max-pooling layer with a series of convolutional filters topped with max-pooling similar to the work in [19]. The effect of adding the BiLSTM to the model in comparison to the bare convolutional model, are reported to be a couple of percentage points in increased testing score.

Chapter 4

Development and Experiments

In this chapter we examine our datasets and explain our experimental architectures. We ended up building 5 different models, 2 based on the cDSSM architecture, and three based on the word2vec embeddings. We will go through some of the issue we had during development, and we will also introduce the baseline method used for result comparisons.

4.1 Datasets

4.1.1 Insurance QA

The InsuranceQA Datsaset are a question and answering dataset retrieved from the website insurancelibrary.com. The questions are generated by consumers with insurance related inquires, and the answers are from experts with deep knowledge of the domain. Some questions contain multiple answers, since many experts can answer the same question.

Question	can i drive a new car home without insurance ?
Answer	probably not . most states lrb including here in ct rrb require proof of insurance as a condition of registering your car . dealerships will require that your insurance professional send them an insurance identification card so that they can process the registration for you . likewise , if you are either leasing or securing a loan to purchase your car , the financial institution will also require you to have the proper insurance coverage in place in order to finalize the sale .

Table 4.1: Example post-reply from the InsuranceQA dataset

In total, the dataset contains 27,413 unique question-answer pairs, where some questions are recurring with different answers. The dataset has already been divided into a training, validation and testing partitions. Average question length is 8.7 word and the average answer length is a 114. The data has all-ready been preprocessed, except for punctuation and special character separation, which were applied by us.

4.1.2 Facebook data

By registering a developer account on Facebook we gained access to their graph API, which allowed us to retrieve all public information by running http queries. We utilized a existing script [**?**] for this task, and adapted it to our needs. We then retrieved all *visitor posts* from the Norwegian telecommunication companies listed in the table below. We excluded post-reply pairs that contained more than 120 words in either message.

Company	Posts retrieved	Posts $<$ 120 words
Telenor Norge	70962	63292
Telia Norge	30141	27961
Chess	18763	17342
Djuice Norge	18382	17052

Table 4.2: Facebook data retrieved

In order to prevent a extensive vocabulary compared to our data-set, we needed to pre-process the data. The following methods were applied.

1. All letters were lower-cased, and then *æ, ø, å* were replaced by *E, O, A*. Afterwards all non-ASCII characters were removed

2. The customers first name, middle name (if present) and last name were replaced with *fname*, *fmname* and *lname*. The customer named were retrieved from the Facebook database along with the posts

3. The name of the company were replaced with the word *companyx*

4. http-links were replaced with the word *weblink*

5. All non-letter and non-number characters outside the set (. , ? ! %) occurring together with letter/numbers were replaced with space. Thereafter all characters in the above set were spaced away from letters and numbers. The result is that each word at the end contains either a mix of letter and numbers or just special characters. This method removed parentheses and brackets around words, and replaced combined word such as *sim-card* with *sim card*.

Orignal question	Bruker dere lang tid på å fikse feil på linje til fast telefon?Har ett bedriftsabonnement (rette tid innen ett døgn sies det), og ett privat abodemang.Begge linjene har det vært feil på siden 21 november. Virker som om det er svært liten vilje til å fikse det. Er det mulig att dere kan få det reparert før jul?
Orignal answer	Hei Einar,Dette er helt avhengig av hva som er årsaken til feilen.Om du ønsker at vi skal sjekke status på feilmeldingen så send oss gjerne referansenummeret i en direktemelding :)/Siw
Pre-processed question	bruker dere lang tid pA A fikse feil pA linje til fast telefon ? har ett bedriftsabonnement rette tid innen ett dOgn sies det , og ett privat abodemang . begge linjene har det vErt feil pA siden 21 november . virker som om det er svErt liten vilje til A fikse det . er det mulig att dere kan fA det reparert fOr jul ?
Pre-processed answer	hei fname , dette er helt avhengig av hva som er Arsaken til feilen . om du Onsker at vi skal sjekke status pA feilmeldingen sA send oss gjerne referansenummeret i en direktemelding :) siw

Table 4.3: Example post-reply from the Telenor Facebook dataset

Since the data contained post-reply pairs of different lengths, we were also interested in observing whether we could produce better results by focusing on data pairs with length below a certain threshold. In addition we were interested in discovering if better results could be obtained if we

were to only focus on a single mobile provider, instead of mixing the data across all companies. After reviewing different partitioning of the data, we decided to split the Facebook data into 3 sets.

Dataset ID	Companies	Post/reply max-length	Remaining posts	Unique words
Tele 1	Telenor, Telia, Chess, Djuice	50 / 90	92446 / 125647	131906
Tele 2	Telenor	30 / 90	29694 / 63292	92959
Tele 3	Telia	60 / 120	23609 / 27961	47248

Table 4.4: Partitioning of the Tele datasets

Next we needed to determine how to deal with low frequency words. For the cDSSM architecture we had two possible solutions. Either drop whole words with frequency below a certain threshold, or only drop tri-letters. By dropping tri-letters instead of words, the system might be more robust to single occurring grammatical mistakes, while reducing the dimensionality by a significant factor. The system may however then be confused to consider two words as equal. After inspecting the words with single frequency, we concluded that removing single occurring trigrams would be the best approach. A large fraction of single occurring words can be categorized are misspellings (and word-concatenations) of words that might be essential to the overall semantics.

For the word2vec architectures random weights were set for words with count below 3. This was simply done by setting the min-count of 3 when training the word vectors.

4.2 Experimental Models

In this section we will explain the model we used for experiments, and the steps necessary before and during training. All models were written in the Python package keras[11], which is a high level library that simplifies the process of building Tensorflow[10] and Theano[28] models with fewer lines of code. Tensorflow and Theano are two separate Deep Learning libraries that allows us to create models by specifying layers and other components as a computational graph, which compiles into C code that can be executed on both CPU and on Nvidia CUDA capable GPU's.

All of our models are implemented as a siamese network, where the question and answer are propagated through two identical networks, and the output is the embedding representation of the sentence. The similarity between the semantic representation for the question are then compared to the semantic representation of the correct answer and a set of negative answers.

4.2.1 cDSSM based models

We started our experimentation by implementing the cDSSM architecture as described in section 3.2. When searching for similar work we found a implementation of the convolution DSSM at[1]. After modifying the code to work with our datasets, we experienced the model to be difficult to train without immediate overfitting. In addition, the model had been written to only work with 1 sample per batch/gradient update, which significantly increased the period of training.

The first step was therefore to adapt the model to work with larger batch-sizes. As far as we could tell, Keras did not have any memory efficient implementation for inputting sparse vectors when using a 1D-convolution layer as input(contrary to the embedding layer that can be used for word2vec). The input had to be converted into a vector of size 3 x tri-letter-size, which resulted in an input size of 120-180 KB per word. This meant that we could not fit the entire training data in memory at once, and it became necessary to write our own generator. This in return gave us some issues when using Theano as backend, where the memory would drain after only few epochs. Unfortunately the Tensorflow backend would not work on our 2GB NVIDIA GTX 670 GPU, so a lot of time was spent for trying to fix the issue. Later in November IDI at NTNU issued us a GPU server, which gave the opportunity to bypass the problem by running the model under Tensorflow at their 12GB Titan X GPU.

The second issue was adopting the model to avoid overfitting, and we were suspicious that it may had something to to with the loss function. In [23], Shen et al implements their query-to-title matching model by using the soft-max function to increase the probability of retrieving the correct document given a query.

$$P(D^+|Q) = \frac{exp(yR(Q, D^+))}{\sum_{D' \in D} exp(yR(Q, D'))} \tag{4.1}$$

In (4.1), R is the relevance/similarity function, y is a smoothing function that was implemented as a learnable parameter, and D is the collecting of the positive document plus J negative documents. This is then used to calculate the loss function by binary cross-entropy (4.2).

$$loss(Q, D^+) = -\log P(D^+|Q) \tag{4.2}$$

In order to push eq. 4.2 to become zero, the optimizer would have to continuously increase R(Q, D+) towards +1 and R(Q, D-) for all negative documents in J towards -1. The problem with this method is that since we are randomly sampling negative examples, the negative document could in

many cases be semantically close to the positive document, but the optimizer would aggressively try to push them as far away in the embedding space as possible. This is our main theory why this loss function did not work.

The solution was to implement different loss function called hinge loss (4.3).

$$loss(Q, D^+) = max(margin - R(Q, D^+) + R(Q, D^-)) \tag{4.3}$$

The advantage of this method is that the objective of the optimizer is to only push the negative similarity away form the positive up until the margin value. In other words, if the margin is set to 0.10, and R(Q, D+) = 0.90, the optimizer will only update the weights if R(Q, D-) is larger than 0.80.

In addition we did some experimentation with the GESD similarity function (4.4) as described in [13], and we found that it outperformed marginally better than the cosine distance function previously used (4.5). We therefore used GESD as our default similarity measure, with c = y = 1.

$$GESD(Q, D) = \frac{1}{1+\|Q - D\|} * \frac{1}{1 + exp(-y(Q * D^T + c))} \tag{4.4}$$

$$cos(Q, D) = \frac{\mathbf{Q} \cdot \mathbf{D}}{\|\mathbf{Q}\|\|\mathbf{D}\|} \tag{4.5}$$

The resulting cDSSM model can be seen in figure 4.1. We tested different choices for the model topology, among them convolution filters = 300, 500, 1000 and output dimension of 128, 300, 500, and the effect of including dropout at different layers. We concluded that a model of 500 convolutional filters, and output embedding dimension at 300 worked best in result-gain per complexity. In addition, dropout only decreased the results significantly, so this was discarded. Weight-sharing between the question and answer module was also tested, and it was observed that this model performed best if the weights were separate for the cDSSM.

The next model we implemented was a cDSSM model with an Bidirectional LSTM layer on top of the convolution layer. The model can be seen in figure 4.2. The purpose of the LSTM layer is to capture semantic dependencies over several trigrams, instead of just extracting the maximum value at each dimension. From the bi-LSTM layer, a k x 400 vector is extracted where k equals the number of word in the propagated sentence. By max-pooling in the word dimension, the model outputs a 400 dimensional embedding vector. For this model, we also experimented if weight-sharing between the question and answer modules were beneficial, and it proved to give shorter training time and better results if the convolutional weights were shared, and the LSTM weights kept separate.

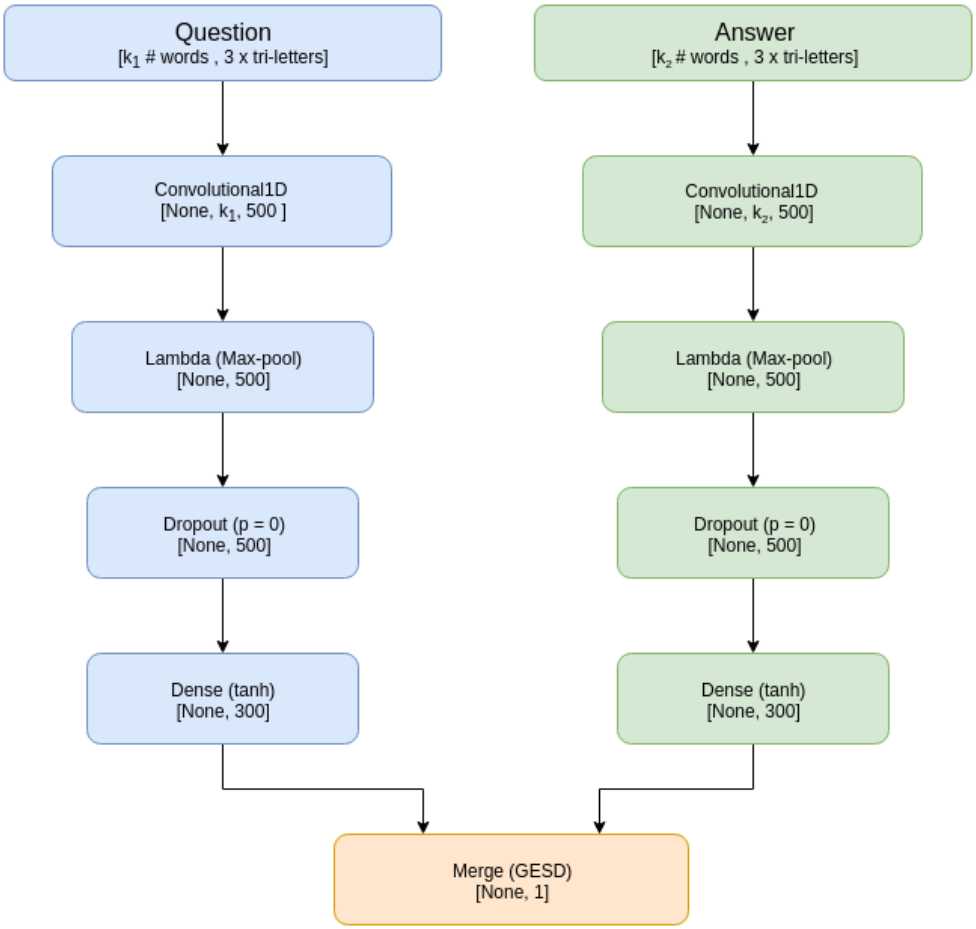

Figure 4.1: Our implementation of the Convolutional deep structured semantic model.

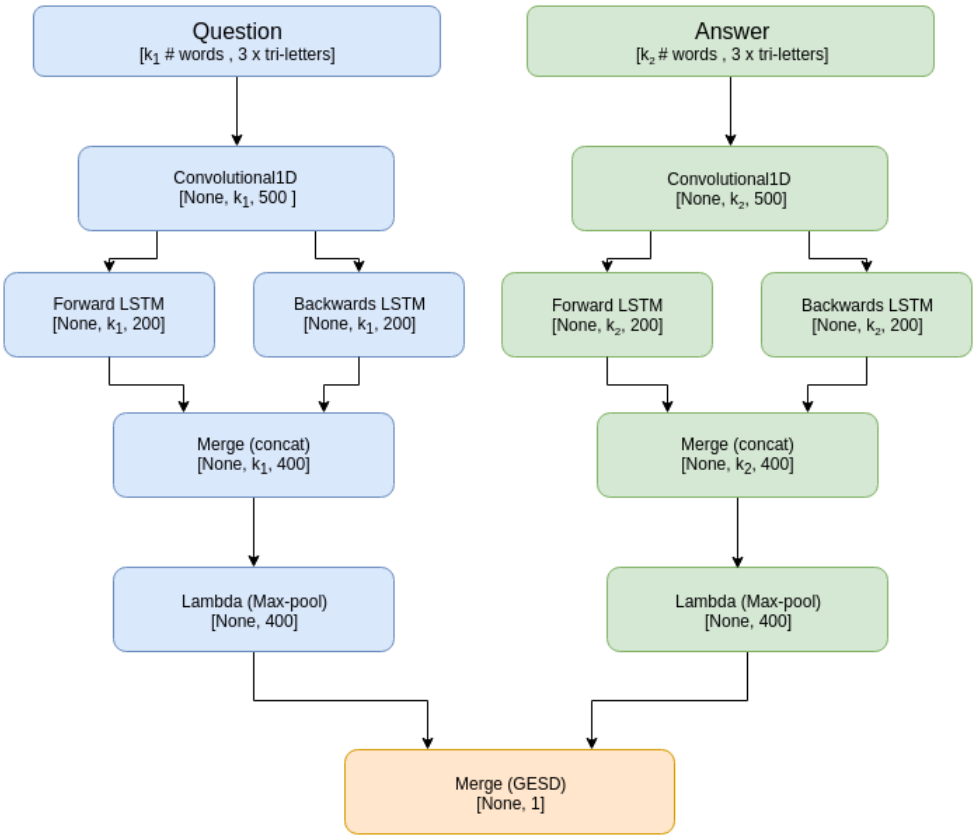

Figure 4.2: cDSSM with Long-short term memory layers.

4.2.2 Word2Vec embedding models

For the Word2Vec based models, we aimed to implement 3 different architectures. By doing some searching we could find a implementation of the models described in [27] on Github [2] and we used this code as our base. The repository included code for generating word2vec embeddings by the python library Gensim citegensim, and some work was done to adopt it for our Facebook data-

The first model we trained was an embedding model(figure 4.3), which has a similar approach as the cDSSM model(word embedding plus single feed forward layer). The already implemented model from [2], included an embedding model without a feed-forward layer, so we made the required alterations. The size of the output were set to be 128.

The second model (figure 4.4) is a convolutional-model, which utilized 500 x 4 filters of sizes 2,3,5 and 7 on top of the word embedding output, which results in a tensor of size k x 2000. The model then performs a max-pooling in the word dimension and redirects the 2000 length result into a feed-forward layer of size 128.

The third model (figur 4.5) utilizes a Bidirectional LSTM and a convolution layer on top. The embedded word representation were used as input to one forward LSTM and one backward LSTM, both with output dimension of 141 per time-step. The resulting output was a k x 282 layer, where as before k is the number of words in the sentence. Then a convolutional layer with 500 x 4 filters of size 1,2,3,5 are added on the bi-LSTM output giving a output of k x 2000, which as in the last model are max-pooled in the word dimension. This gives us a embedding space of 2000 dimensions.

As for the cDSSM models we utilized the Hinge-loss function (4.3) as cost function for all 3 architectures. For similarity function we initially used the GESD (eq 4.4), but we had some issue with numerical stability and resulting NaN values, and it was therefore replaced with the cosine function (eq 4.5)

Before we could start training our models, we needed to determine which Word2Vec parameters to use. This included whether to use CBOW or skip-gram, which dimension-size, and which window-size. We also needed to determine if we would use embeddings trained on the entire corpus, or embedding trained on the specific training set, and if the embedding weight were to be set non-trainable when learning the Q/A model. Locking the embedding-weights can in theory yield better performance, due to it will force the optimizer to only learn the LSTM and convolution weights.

We ran some experiments with different parameters, and we made the following observations.

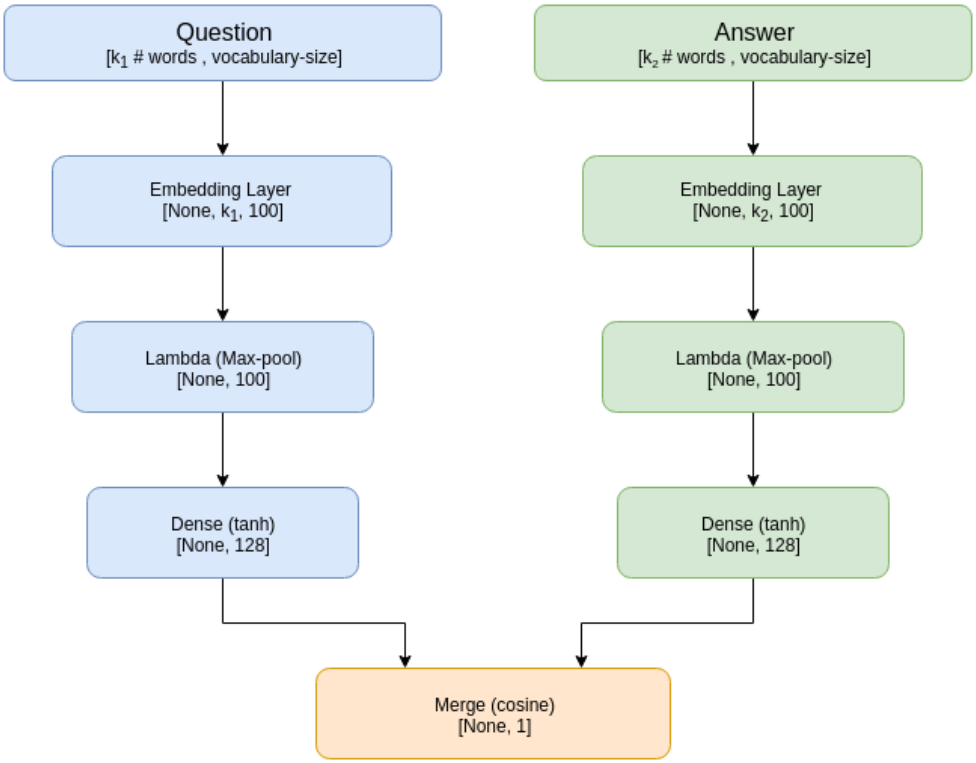

Figure 4.3: Word2vec embedinng model. The weights for the embedding and dense layers are shared between the question and answer parts.

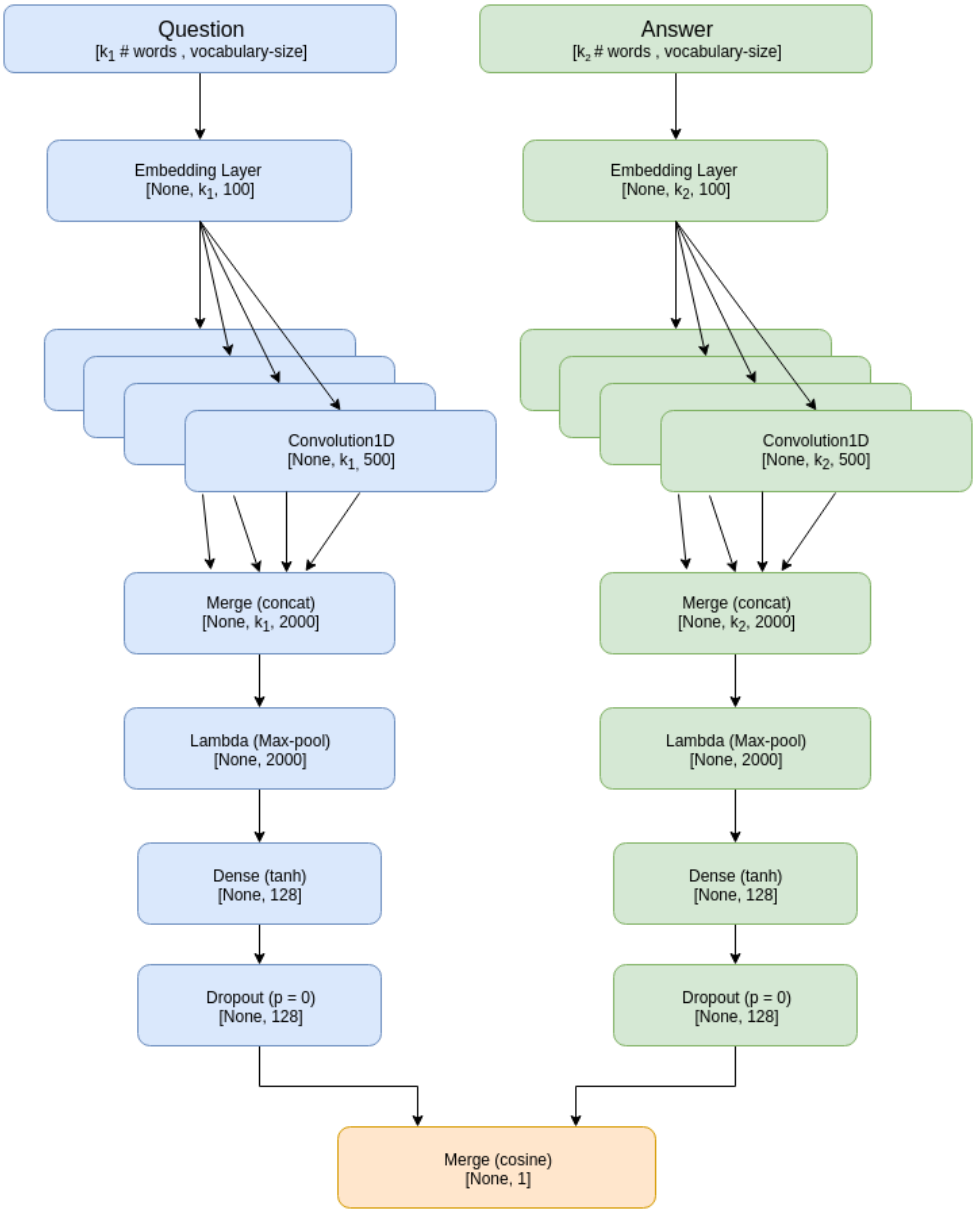

Figure 4.4: Word2vec model with convolutional filters. The weights for the embedding layers, convolutionals layers and dense layers are shared between the question and answer parts.

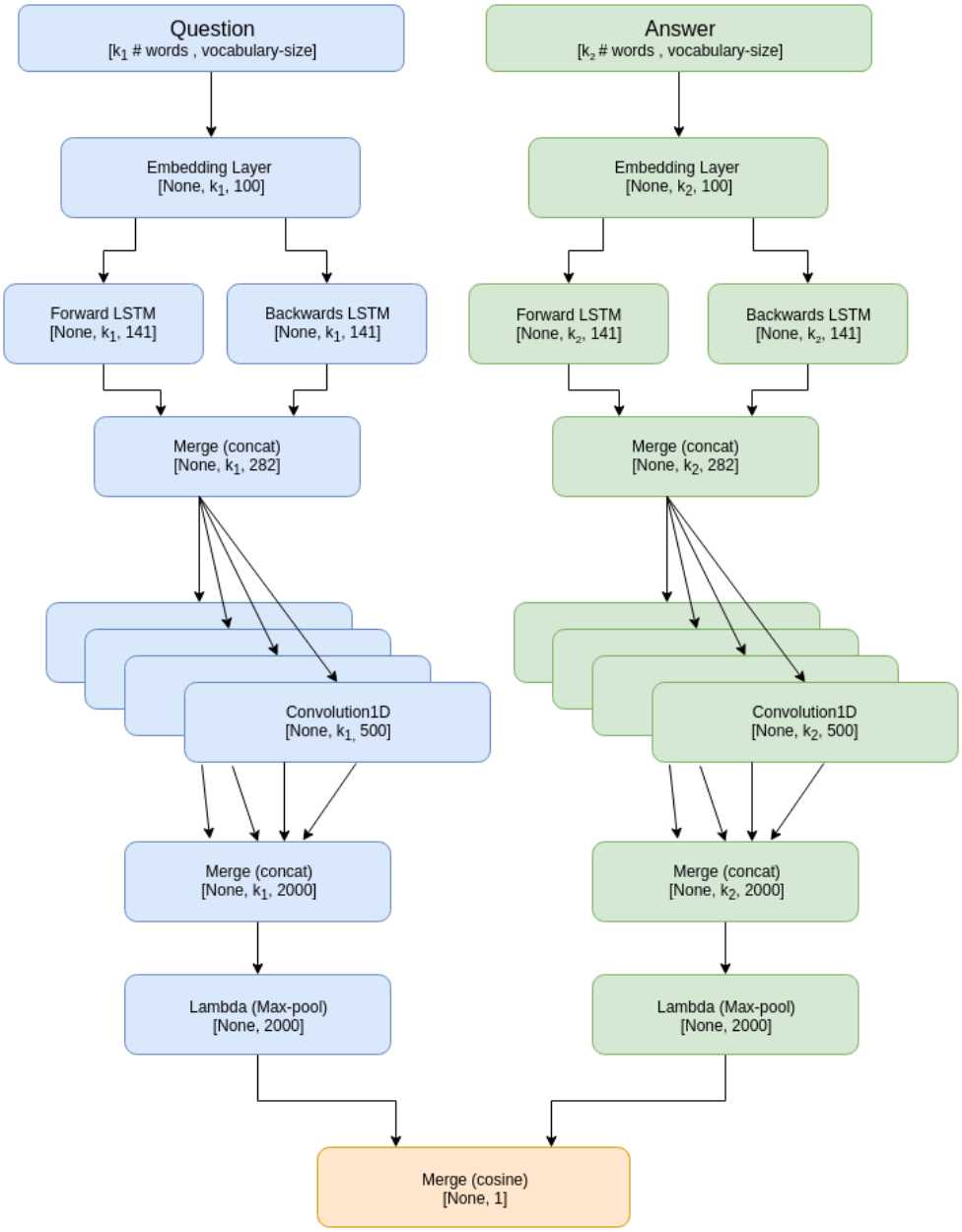

Figure 4.5: Word2vec model with LSTM and convolutional filters. The question and answer weights are also shared in this model.

1. Embeddings trained on the training set were better than training on our entire corpus.
2. Non-trainable word embeddings gave faster convergence, but worse overall results.
3. Skip-gram outperformed CBOW, and dimension of size 100 were preferable over dimension size 300 with shorter training time and slightly better results. Window size of 8 yielded better results than window size of 5 and 3.

4.3 Baseline methods

In order to validate the effectiveness for our neural models, we needed a baseline model for comparison. We decided to use the BM25 ranking model, which is often used for information retrieval tasks. In order to achieve the optimal results, we needed to preprocessed the data, and we applied the following steps.

1. Tokenization - Non character letter were removed, by using the NLTK [7] package. This were originally designed for English text, but it also worked well for our Norwegian dataset
2. Stop-word removal - Stop-words such as 'is, to, you, be' etc were removed by using the python package stop-words [9]. It also includes stop-words for Norwegian which we used on our Facebook data.
3. Stemming - Each word were reduced to the root of the word. By example: stemming, stemmed, stemmer, stems were all reduced to stem. We utilized the NLTK.stemmer package for this task, which contains stemmers for both English and Norwegian.

For each post in the testing set, we randomly selected 500 replies including the correct response, and applied the BM25 algorithm to rank each of the candidate replies given the post.

We also experimented by implementing an LDA topic matching model[6], by using the gensim libary. We created a document corpus of all post/replies in the testing set (posts and replies were two separate documents), and then applied the LDA algorithm to generate topics vectors for the entire corpus. We then used the topic distribution for each post to find the most appropriate reply. Unfortunately we could not get any significant results above random for our datasets, by this method.

Chapter 5

Results and Analysis

In this chapter we will revise our experiments and give an overview of the results. We will begin by giving some details about the experimental setup, including the hardware, time and which parameters that were used in order to produce the results. Thereafter we will go through a quantitative analysis of the results, were we will introduce our metrics and the results for each model per dataset. Then we will perform a objective evaluation of some of the best performing models, to see if the model actually manages to capture the semantic complexity.

5.1 Experimental Setup

Our development phase as explained in Chapter 4 leaved us with 5 individual models that were to be trained on 4 different datasets. For any model that utilized convolutional layers, a machine with a GPU was essential. The models were also needed to be trained for several hours in order to find usable parameters and fix issues, so in effort of reducing the total training time, we ran our models on 3 different systems:

- Samuel server issued by IDI. The machine contained an Intel 5930k 12 core CPU, 2x Titan X 12GB GPU's and 64 GB of system RAM. It is however shared between several students, so models could only be deployed when there was free capacity.
- Personal Desktop of the author. Featured a Intel 5820k 12 core CPU, 1x GTX 1060 6GB GPU and 12 GB of system RAM. This is machine was available 24/7, and models were almost

run continuously during the experimentation period.

- Amazon AWS p2x instances [4]. For roughly 30 US cents per hour, instances featuring a shared Xeon CPU, shared NVIDIA Tesla k80 12GB GPU and 64 GB of system RAM could be rented in the cloud. We ended up deploying two of these during training of the final cDSSM models.

We partitioned the data into training, validation and test sets. The training set is the samples which the model are tuned on, the validation set exists for testing how well the model generalizes on unseen data, and the test-set is for overall confirmation of the model's capabilities. The split is detailed in the table below.

Id	Details	Training	Validation	Test
Ins	InsruanceQA	14800	3700	3000
Tele 1	All companies, max-length 50-90	70000	17500	3000
Tele 2	Telenor, max-length 30-90	21600	5400	1800
Tele 3	Telia, max-length 60-120	17200	4300	2000

Table 5.1: Dataset split

Before we could start training our models, an appropriate value for the hinge-loss margin would have to be found. In [27] it was reported that a margin between 0.05 and 0.20 gave decent results, so we mainly focused on exploring values in this interval. The optimal value for each model can be seen in table 5.2. In addition we needed to determine which optimizer to use for handling the gradient updates. A common optimizer is the Stochastic Gradient Decent[8] algorithm, however this requires tuning of many hyper-parameters such as learning-rate, momentum and decay which would entail many more hours of training. A simpler approach is to use an adaptive optimizer, such as adam[20], adagrad[12], and adadelta[31], and we mainly used adam or adadelta for our experiments. In order to prevent overfitting we employed a simple strategy called early-stopping, which basically means we stop the training after a number of epochs has passed without improvements on the validation data. The patience was set to 5 for most models, and we always kept the model with the lowest validation loss. The models were trained by mini-batches with size 25-100 dependent on the model and dataset.

Model	Dataset	Hinge margin	Machine	Time/epoch	Best epoch
cDSSM	Ins	0.08	Desktop	15 min	15
cDSSM	Tele 1	0.08	Samuel	55 min	20
cDSSM	Tele 2	0.08	Samuel	13 min	19
cDSSM	Tele 3	0.08	AWS p2	16 min	17
cDSSM-LSTM	Ins	0.10	Desktop	12 min	24
cDSSM-LSTM	Tele 1	0.10	Samuel	110 min	19
cDSSM-LSTM	Tele 2	0.10	Samuel	45 min	24
cDSSM-LSTM	Tele 3	0.10	AWS p2	35 min	18
w2v-embedding	Ins	0.05	Desktop	2 sec	72
w2v-embedding	Tele 1	0.05	Desktop	30 sec	10
w2v-embedding	Tele 2	0.05	Desktop	5 sec	32
w2v-embedding	Tele 3	0.05	Desktop	5 sec	30
w2v-convolution	Ins	0.15	Desktop	30 sec	11
w2v-convolution	Tele 1	0.15	Desktop	8 min	12
w2v-convolution	Tele 2	0.15	Desktop	1 min	6
w2v-convolution	Tele 3	0.15	Desktop	41 sec	9
w2v-LSTM-conv	Ins	0.05	Desktop	112 sec	?
w2v-LSTM-conv	Tele 1	0.05	Desktop	8 min	12
w2v-LSTM-conv	Tele 2	0.05	Desktop	3 min	26
w2v-LSTM-conv	Tele 3	0.05	Desktop	2.5 min	12

Table 5.2: Training details

5.2 Quantitative Results

We will now present our performance metrics, which are metrics often used for document ranking problems. The first is the Mean Mecirproal Rank, which gives an average over the inverse rank for each target document. Given a set of queries Q and their corresponding documents D+, MMR can be calculated by:

$$MMR(Q, D+) = \frac{1}{|Q|}\sum^{Q}\frac{1}{rank(D+)} \tag{5.1}$$

The second is the TopK score, which tells us how often the correct document occurs within the k

first documents.

$$Top(Q, D+, k) = \frac{1}{|Q|}\sum^{Q} \begin{cases} 1, & \text{if } rank(D+) \geq k \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

We performed these evaluations by comparing the similarity for each question in the test-dataset with the similarity for the correct answer, and 499 randomly sampled answers. The following tables contains the scores for the different datasets and experimental models.

Model	MMR	Top1	Top5	Top20
BM25	0.368	0.257	0.471	0.674
cDSSM	0.295	0.160	0.421	0.757
cDSSM-LSTM	0.459	0.342	0.577	0.841
w2v-embedding	0.500	0.356	0.683	0.884
w2v-convolution	0.598	0.451	0.784	0.950
w2v-LSTM-conv	0.652	0.525	0.808	0.956

Table 5.3: Results for the InsraunceQA dataset.

Model	MMR	Top1	Top5	Top20
BM25	0.153	0.082	0.203	0.371
cDSSM	0.280	0.147	0.406	0.729
cDSSM-LSTM	0.186	0.083	0.258	0.537
w2v-embedding	0.238	0.146	0.321	0.542
w2v-convolution	0.268	0.157	0.371	0.643
w2v-LSTM-conv	0.320	0.202	0.443	0.704

Table 5.4: Results for the Tele 1 dataset .
All companies, max-length 50-90

Model	MMR	Top1	Top5	Top20
BM25	0.175	0.092	0.217	0.385
cDSSM	0.239	0.124	0.334	0.622
cDSSM-LSTM	0.154	0.066	0.219	0.483
w2v-embedding	0.202	0.119	0.283	0.487
w2v-convolution	0.234	0.131	0.328	0.595
w2v-LSTM-conv	0.277	0.176	0.379	0.618

Table 5.5: Scores for Tele 2 dataset.
Telenor, max-length 30-90

Model	MMR	Top1	Top5	Top20
BM25	0.193	0.115	0.240	0.402
cDSSM	0.147	0.067	0.196	0.460
cDSSM-LSTM	0.143	0.067	0.201	0.433
w2v-embedding	0.184	0.107	0.246	0.452
w2v-convolution	0.224	0.120	0.318	0.586
w2v-LSTM-conv	0.266	0.165	0.360	0.618

Table 5.6: Scores for Tele 3 dataset
Telia, max-length 60-120

Overall we can see that our best performing architecture was the Word2Vec-LSTM + convolution based model. Except for a single instance (Top20 for the Tele 1 dataset), this model outperformed all the other architectures in all our tests. The cDSSSM model did however prove itself useful for the tele-datasets, outperforming the w2v embedding and w2v convolutional model. For the InsuranceQA dataset, it did not perform as well though, and we can conclude that the cDSSM-LSTM was a better fit for this task.

5.3 Semantic Evaluation

By again reviewing our immediate results from the last section, we can observe that splitting the Facebook data into smaller(company specific and length limited) partitions were not beneficial to the overall performance. This was most likely due to the decrease in training data, were we at the same time had to deal with the fact that the diversity and complexity of the corpus stayed almost the same. We will therefore not include results for these models in our further analyses.

In tables 5.7 to table 5.10 we present sample outputs from our best performing models from the cDSSM and word2vec architectures. Our goal is now to give some understanding whether our models is capable of capture the semantic context regardless of the true answer ranking score.

1. Question	how important is disability insurance ?
1. Candidate Answer	modified whole life insurance is really a company specific question . for one of the companies i work with , modified whole life is designed for someone who has a medical history which most companies would...
1. True Answer	how important is having a paycheck ? try to imagine no paycheck . how much money would you need to pay your bills if you were disabled 30 months , the average time of a disability ? so buy term and spend...
2. Question	does home insurance cover windows ?
2. Candidate Answer	most life insurance companies will not reinstate a life insurance policy after surrender . once a cash value is returned to you for permanent coverage , the life insurance company will declare themselves free of liability...
2. True Answer	if your window is damaged or broken and it is caused by a covered peril in your policy , then yes , you will have coverage , of course subject to your deductible . keep in mind that you need to be residing in your home...
3. Question	can you go to jail for not having auto insurance ?
3. Candidate Answer	in most states , insuring your car or being financially responsible is a requirement ofoperatinga vehicle . if you drive without these types of coverage ...
3. True Answer	in most states , insuring your car or being financially responsible is a requirement ofoperatinga vehicle . if you drive without these types of coverage...
4. Question	how long does disability insurance last ?
4. Candidate Answer	rentersinsurance covers the personal property of an individual or family that does not own the dwelling that they occupy . therefore , the contents of the building lrb often an apartment building rrb will need insurance...
4. True Answer	it can last from 6 months to 1 year depending on which policy to signed up for . if it is short term disability it will last 6 months and long term 1 year . always ask your agent to explain which plans they are discussing...
5. Question	which retirement plan is best ?
5 Candidate Answer	car insurance can go up for women just like they can go up for men . there is no difference really other then women in general get a slightly lower rate if all things remain equal . other then that the basic same rules...
5. True Answer	which retirement plan is best ? the one that you actually do . it is more about the habit of saving than anything else . developing a savings habit will aide anyone in retirement . waiting for the optimal plan before you...

Table 5.7: Sample of questions and responses from the InsuranceQA data generated by the cDSSM-LSTM model

1. Question	be renter insurance mandatory in Michigan
1. Candidate Answer	no it will not under every renter insurance policy that I be aware of renter insurance cover the personal property of the insure immediate family that reside in the same residence only protect yourself and...
1. True Answer	renter insurance be not mandatory , however it be a good idea have affordable your landlord policy only cover the structure and not your personal item it may even help reduce your auto policy by have...
2. Question	what do a qualified retirement plan mean
2. Candidate Answer	what do a qualified retirement plan mean it mean a retirement plan that offer some sort of tax benefit the participant in their
2. True Answer	what do a qualified retirement plan mean it mean a retirement plan that offer some sort of tax benefit the participant in their
3. Question	can you get Life Insurance without a physical
3. Candidate Answer	the most life insurance that I know of that you can get without a physical in 1 policy be 500,000 this be at certain age and within health parameter person too young old get this much or who have poor health may...
3. True Answer	yes get life insurance without a physical can be a very good way get life insurance especially if you do not fit in the very good health class due weight minor health condition people sometimes opt take an exam...
4. Question	how much do Medicare cover for Mri
4. Candidate Answer	good question Medicare normally cover drug under part b in very limited situation such as injection administer in a doctor office , certain oral cancer medication drug use with durable medical equipment such...
4. True Answer	generally speak Medicare cover 80 percent any service that be deem medically necessary your doctor or hospital if you just have Medicare part A and Medicare part b you will be responsible for the 147.00...
5. Question	can you drive any car with fully Comp insurance
5 Candidate Answer	if you be speak of your vehicle while you be drive striking a bicyclist then yes your automobile insurance will typically cover the damage if you be ask from the stand point you be the bicyclist and be involved...
5. True Answer	automobile insurance cover the car if you have an automobile policy and be operate someone elses automobile , you may have some coverage , shall the policy cover that auto be inadequate it be difficult...

Table 5.8: Sample of questions and responses from the InsuranceQA data generated by the word2vec-LSTM-conv based model

By looking at the output form the InsuranceQA trained models, we fist observe that the cDSSM-LSTM model produced significantly worse results than the Word2vec based model. In fact, in our sample, the results retrieved were relatively far from semantically coherent with the question and true answer. Except for the case where the model predicted the correct answer, the model was mostly retrieving answers not related to current questions at all. At some part we would have to assume that we were unlucky with our sample, but also it should be clear the cDSSM is not a good model choice for this task. The Word2vec based model on the other hand produced almost excellent results. Although it did not retrieve the true factoid answer for each question, all replies were semantically relevant and they address the issues for each query.

1. Question	hvordan er det me fri bruk for mitt abonnement nå ? jeg fikk 3 mnd fri bruk når jeg kjøpte min tlf i høst .
1. Candidate Answer	hei fname . det trenger du ikke . send oss mobilnummeret ditt i en direktemelding så fikser vi det :) siw
1. True Answer	send oss nummeret ditt i direktemelding så skal vi undersøke nærmere . morten
2. Question	heisann ! er nettet nede i hagebyen trondenes i harstad i kveld , eller er det bare hos meg det er en stor muffins ..?
2. Candidate Answer	hei fname . det er ingen kjente feil i området . er det flere enn deg som opplever problemer med dekninga ? har du prøvd å slå mobiltelefonen av og på ? :) jan olav
2. True Answer	hei fname . det er ingen kjente feil i området . er det flere enn deg som opplever problemer med dekninga ? har du prøvd å slå mobiltelefonen av og på ? :) jan olav
3. Question	hei . jeg sendte dere direktemelding for noen dager siden , som i følge deres nettsider skulle gå raskere enn mail . hvor lang responstid har dere egentlig ? mvh annette
3. Candidate Answer	hei fname . uff da , har du ikke mottatt svaret vårt ? vi svarte deg på direktemeldingen for ca 6 timer siden , svaret finner du på samme sted som du sender henvendelsen . vi kan ikke se å ha...
3. True Answer	hei ! vi svarer opp alle henvendelser her på facebook fortløpende , men vi kan ikke se å ha mottatt noen direktemelding fra deg disse siste dagene . :/ sjekk gjerne at meldingen du sendte har...
4. Question	takker companyx nok en gang . enkelte av dere er bare helt rå :)
4. Candidate Answer	hei fname . det er ikke noen planlagt nedetid som vi er informert om . det er klart at disse tingene blir varslet om og normalt vil vedlikeholds vinduer bli lagt til tider med lite trafikk . sånn type...
4. True Answer	takk for det , fname . du er også rå ! :) ha en fin dag ! anja
5. Question	spørsmål om dekning i varntresk . hattfjelldal kommune . gjelder området varntresk nordalsbukta . dekning er blitt betraktelig dårligere siste år .
5 Candidate Answer	hei fname mname ! der kan det nok være noe svak dekning per nå , men det er planlagt en utbedring der i løpet av de neste månedene . mvh andreas
5. True Answer	hei finn ove :) dette må vi få undersøkt . ser at dekningen i utgangspunktet ikke er den aller beste der . er det nå slik at du ofte mister dekningen helt ? gjelder dette kun deg eller vet du om...

Table 5.9: Sample ranking from the cDSSM for the FaceBook Tele 1 dataset

1. Question	hei , jeg er i utlandet og fikk åpnet abonnementet mitt igjen i dag . men jeg har fremdeles ikke dekning . kan dere sjekke hva problemet er . nummeret mitt er 97895534 . jeg er på lanzarote .
1. Candidate Answer	hei fname , dette må vi sjekke for deg : vi tok en titt på nummeret ditt her hos oss og dette har blitt flyttet over til ny leverandør i dag , derfor vil du ikke lengre ha dekning på sim kortet fra oss . har du satt sim kortet...
1. True Answer	hei fname , det stemmer . for opplysningens skyld , gjør vi deg oppmerksom på at ved en sånn prosess kan det ta opptil to virketimer før du får dekningen tilbake . vi anbefaler deg å skru av og på telefonen , sånn...
2. Question	hei ! vil dere få inn samsung ativ s ?
2. Candidate Answer	hei fname mname :) vi har dessverre ingen konkret informasjon om når samsung galaxy note eventuelt kommer til vår nettbutikk . oscar
2. True Answer	hei fname . vi har dessverre ikke hørt noe om når den eventuelt kommer dessverre . bjørnar
3. Question	eg har micro simkort og skulle hatt nano asap. er det mulig og få dette ut i div butikker eller må eg få i posten fra dere ?
3. Candidate Answer	hei fname . den raskeste måten er å gå innom en forhandler . da får du simkortet på dagen . :) per reidar
3. True Answer	hei fname . du kan kjøpe nytt sim kort hos en forhandler som selger våre abonnementer , det koster opp mot kr 199 ,- og da får du det i hånden og kan ta det i bruk med en gang :) kim
4. Question	hei, bestilte det nye abonnementet. kan jeg begynne å bruke det med en gang?
4. Candidate Answer	heisann ! send oss en direktemelding med mobilnummeret ditt og fødselsdato , så kan vi sende dette . du kan goså enkelt bestille dette på mine sider :) miriam
4. True Answer	hei håkon . gjelder det nummeret som slutter på xxx xx x80 ? isåfall er denne oppdatert , så det er bare å surfe i vei . :) fahd
5. Question	hei . jeg kjøpte ny mobil med companyx l for noen måneder siden , men 4g dekningen er kjempe dårlig ! det tar en evighet å komme inn på sider nesten uansett hvor jeg er . og 4g bare buffrer hele tiden ...
5 Candidate Answer	hei fname . det kan nesten høres ut som at mobilen prøver å koble til 4g der det er svak eller ingen dekning . prøv gjerne å deaktivere 4g når du opplever dette igjen og se om det gjør en forskjell . morten
5. True Answer	hei fname . dette må vi få undersøkt : hvordan opplever du dekningen ? er det mobildataen det gjelder , eller påvirker også dette samtaler ? kunne du forsøkt å skru av 4g , og kun låse mobilen til 2 3g da . hvordan...

Table 5.10: Sample ranking from the Word2Vec-lstm-conv model

In table 5.9 we can see the results produced by the cDSSM model for the Tele 1 dataset. The first question were not answered with a semantically relevant reply, but the model managed to capture what action the customer should take(send a direct message). The second question were answered by retrieving the correct answer. For the third question the model managed to produce a reply that were semantically correct, but the facts were wrong. Here the model generated reply assumed that the message had already been answered, instead of the true situation where the customer service agent had not registered the previous customer message. Question 4 were both factually wrong and semantically wrong (talking about downtime when the customer tried to express gratitude). A hypotheses for why this was response was generated, is that the data contains much sarcasm (by example "No signal today either. Thank you so much!"), which confuses the model. The 5. question were also semantically relevant, but as for many of the other posts it references a real world event, and it cannot understand the importance of the details(in this case the location).

Next we will go through the results for the word2vec-lstm+convolutional model, and the post-reply pairs can be seen in table 5.10. The first question were answered with a reply that is semantically close, and it here manged to reference the issue the customer was facing(new mobile provider, and no reception), however the model assumes that the customer had left the company, instead of just signed up with them. The second question were regarding at which time a certain mobile phone will be in stock, and the model retrieved an answer with the same context(not in stock) but the it also referenced a different model from the same brand. Question number 3 is regarding getting a new sim-card, and while it did not retrieve the correct answer, the model refereed the customer towards the correct action(go to one of the companies stores and pick one up). Quite impressive. The 4. question were not answered with a semantically close answer, and simply it is simply telling the customer to contact them, when it should have told the customer that the subscription was ready for use. For the last question, the model managed to capture the issue(problem with the 4g connection), but it is a bit unclear whether the candidate answer actually addresses the issue since it is telling the customer to deactivate the 4g to fix the problem. However there is obviously some semantic relation, although the solution and problem understanding might be different.

5.4 Clustering and Visualization

Another method for analyzing the results, are by visualizing the embedding space by a technique called Distributed Stochastic Neighbor Embedding(t-SNE) invented by Maaten et al in 2009 [21]. By grouping together data points that are distance-wise close in the original embedding space, the

algorithm enables us to project the data a space with only 2 or 3 dimensions, thereby making it human readable.

In order to add some meaning to the visualization, we needed to figure out a way to classify our different posts. Our original idea was as with the baseline method to use Latent Dirichlet allocation, which would enabled us to generate a set of topics for our dataset together with the topic distribution for each post. However, we were not successful of retrieving any meaningful topics by this method, and after some effort the idea was discarded. Instead we simply looked at the data an observed some repeating keywords, and then classified each post by the keyword as the label given that it contained the word(or part of the word). We decided to use the cDSSM model with the Tele1 data-sett for this task, and the visualization can been seen in figure 5.1.

As we can see for scatter plot, there is a somewhat clear clustering tendency for our identifiable classes, sim-kort (sim-card), dekning (reception), faktura (billing) and direktemelding (direkt messaging). Given a better classifying method, we should be able to detect more patterns.

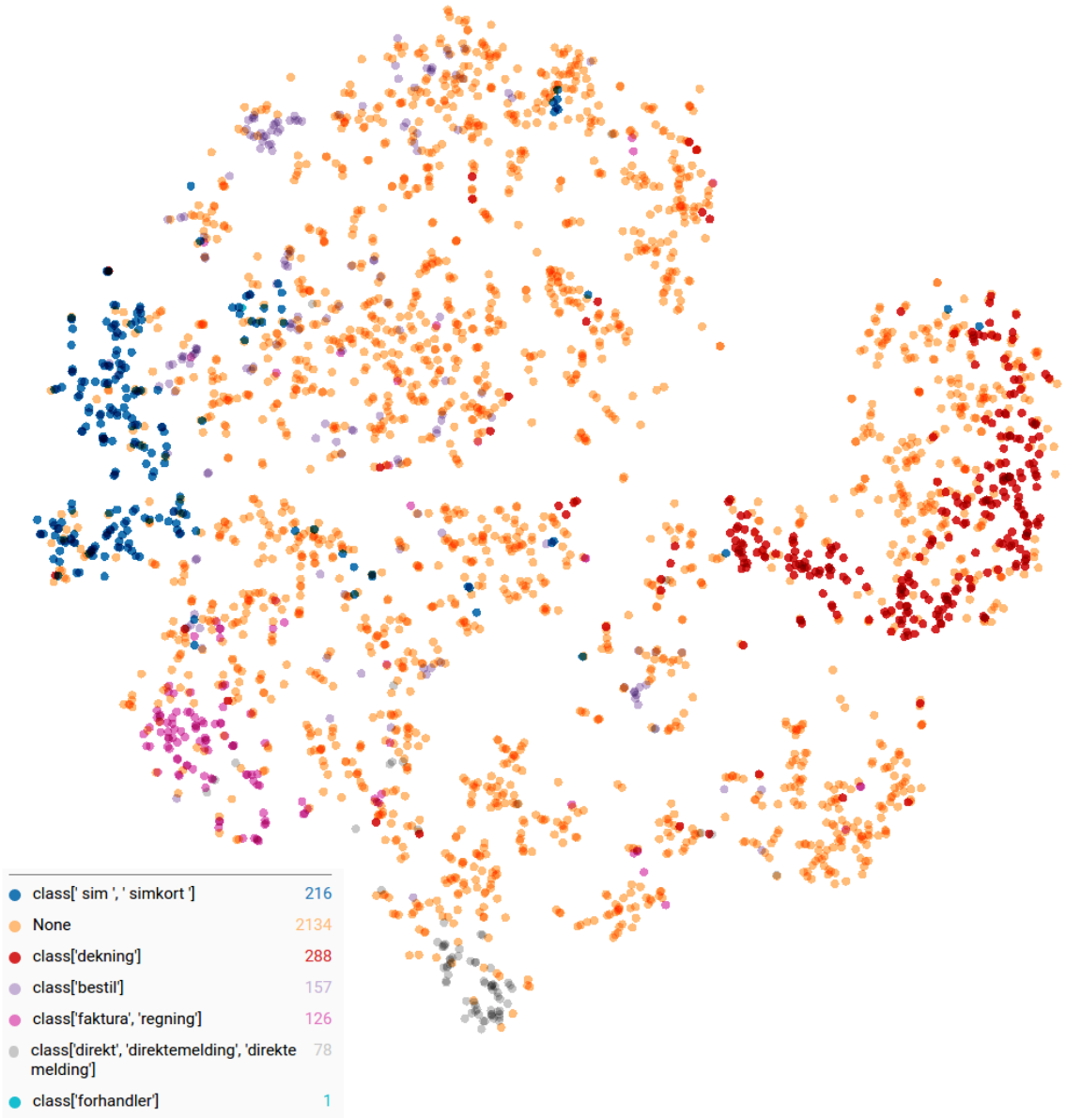

Figure 5.1: t-SNE plot of subset of the Tele1 dataset.

Chapter 6

Summary and Further Work

6.1 Project Summary

During this project we have investigated different approaches for building a retrieval based question/answering system by using neural networks. We started the project by reviewing state-of-the-art research, including Deep Structered Semantic Models from Microsoft Research, word2vec embeddings, and Bidirectional LSTM models from the IBM Watson group.

After a lengthy development period, we managed to test the system on 2 different sources of data, the InsuranceQA dataset and social media data retrieved from Facebook. The Facebook data was further split into three subsets. The first subset (referenced as Tele1) contained posts from all companies, and samples of post-replies width lengths above 50/90 words were removed. The second dataset (Tele2) contained only post from the Facebook page of Telenor AS, and samples were pruned by maximum lengths of 30 and 90. The third dataset (Tele3) contained posts from the Facebook page of Telia AS, and the maximum allowable length for each post/reply were set to 60/120 words.

We implemented a total of five different models, in addition to the BM25 ranking model for baseline comparison. Our first two models (refereed to as cDSSM and cDSSM-LSTM) were built on the Convolutional Deep Structured Semantic architecture, first proposed in [23]. The common core of both models are that each sentence is inputted as a collection of trigrams, and each word in the trigram are represented by a tri-letter hashing function. By performing one dimensional convolution

on the trigrams, the model extracts the semantic representation of the three words. For the cDSSM model, the output from the convolutional filters are then connected to a word-vise max-pooling layer, and thereafter a feed forward layer resulting in a one dimensional semantic vector representation. For the cDSSM-LSTM, the convolutional output was connected to a bi-directional LSTM recurrent neural network, where each trigram was inputted at each time-step.

The other three models were based on the skip-gram Word2Vec word representation[22], which were pre-trained by using the gensim[5] python library. The w2v-embedding model utilized a max-pooling layer on top of the word embeddings, connected to a feed-forward layer for semantic encoding. The second model (w2v-convolutional) employed a series of convolutional layers with different filter lengths to extract semantic representation over several words, then connected to a max-pooling layer in the word dimension, topped with a final feed forward layer to reduce the dimensionality. The last model, referenced as w2v-LSTM-conv, utilized a bidirectional LSTM layer, combined with a series of convolutional filters and at the top a word-vise max-pooling layer.

After training the models, we measured their performance by calculating Mean Reciprocal Rank and TopK scores. We observed that the overall best performing model was the w2v-LSTM-conv on all datasets and subsets. This model was also significantly faster to train than the cDSSM models, which leaves us to conclude that the word2vec embedding approach is significantly preferable over the cDSSM approach. We can make this conclusion by looking at both the quantitative scores and the semantic evaluation presented in section 5.3.

6.2 Discussion

The results leaves us with the question: Why did the models perform as they did? Most likely the answer can be found in our datasets. As explained in chapter 4, the InsuranceQA dataset is in many ways a simple dataset in terms of grammatical correctness and relatively short questions. In addition, it is presumable to say that the topics are more factual based and temporal static, where the dataset has been collected within a short time period(1-2 years), and the domain(laws regrading insurance) will most likely not have clanged significantly within the sample time period.

The Facebook dataset is on the other hand highly temporally depended and are often referencing entities such as products or locations with low word frequency. Take the question "Do you have Iphone 6s in stock?". This is a question that depends on the real-world state of the company inventory, and in the case where there has been many such inquiries during a out-of-stock period, the model will be trained to answers such questions with a negative reply, even if the product at

query time is in stock. The same cane be said for locations where questions such as "Is the reception down in Ranheim?" can be hard to retrieve correctly.

An additional challenge is for the model to understand the importance of such entities when they have a single occurrence. It might be able to capture repeating phrases, but without enough training data it will be difficult for the architecture to learn the importance of low frequency(but important) entities, and at the same time discard other words such as misspellings etc. During training it was observed that the model gave better semantic results, when questions or answers were less specific, such as "please contact me", "thank you for the great service" etc.

If we go back the the results, we can observe that our models performed significantly better for the Tele1 training data, compared to the smaller Tele2 and Tele3 datasets. This give us an indication that more training data would be needed for better results, and preferably combined with solutions to the temporal and entity problems discussed above. It is also worth mentioning that most literature available for the cDSSM architecture has been research where the models has been trained on millions of samples, and mostly on data with significant shorter sentences.

6.3 Future Work

As we have seen with our results, developing a customer-service dialog system by using only the techniques described in this rapport, is far from optimal. Trying to answer factual time dependent questions with a retrieval system, is obviously difficult without some reasoning tool and look-up capabilities on the company knowledge base. An interesting approach would be to adapt something similar to semantic parsing as discussed in [30], where they use the cDSSM model to match sentences to the entities mentioned and the relation pattern. In this manner for our sentence "is the reception down in Ranheim?", the model could extract the entity "Ranhem" and the relation "reception down in-?".

Another approach would be to add information from the company knowledge base directly as context to each massage when training the model. In [29] Yan et al. successfully trains a model where they use the previous message in the conversation as context in a multi-turn chat environment. Their model works by creating a new query, concatenating the original message with the context. Their model is relatively similar to our w2v-LSTM-conv model. Given that we are able to retrieve the relevant information from the knowledge base upon query time, the model could add information about the product/location/etc as context.

Both these approaches requires however that we have access to datasets with corresponding knowledge bases, where historic data also is available so we can extract the background information at the time the training sample was produced. This is however not easy to obtain, since such data often is guarded by privacy rules and internal business secrets.

Bibliography

[1] Github airalcorn2. https://github.com/airalcorn2/Deep-Semantic-Similarity-Model. Accessed: 2016-12-21.

[2] Github codekansas. https://github.com/codekansas/keras-language-modeling. Accessed: 2016-12-21.

[3] Stanford glove. http://nlp.stanford.edu/projects/glove/. Accessed: 2016-12-21.

[4] Amazon web service. https://aws.amazon.com/ec2/instance-types/. Accessed: 2016-12-21.

[5] Gensim topic modeling. https://radimrehurek.com/gensim/. Accessed: 2016-12-21.

[6] Latent dirichlet allocation. https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation. Accessed: 2016-12-21.

[7] Nltk python packageg. http://www.nltk.org/. Accessed: 2016-12-21.

[8] Stohastic gradient decent. https://en.wikipedia.org/wiki/Stochastic_gradient_descent. Accessed: 2016-12-21.

[9] Stop words python. https://pypi.python.org/pypi/stop-words. Accessed: 2016-12-21.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[11] François Chollet. keras. https://github.com/fchollet/keras, 2015.

[12] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. URL http://dl.acm.org/citation.cfm?id=2021068.

[13] Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 813–820, 2015. doi: 10.1109/ASRU.2015.7404872. URL http://dx.doi.org/10.1109/ASRU.2015.7404872.

[14] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Neural Information Processing Systems (NIPS)*, 2013.

[15] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 699–709, 2014. URL http://aclweb.org/anthology/P/P14/P14-1066.pdf.

[16] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. Modeling interestingness with deep neural networks. In *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 2–13, 2014. URL http://aclweb.org/anthology/D/D14/D14-1002.pdf.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[18] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 2333–2338, 2013. doi: 10.1145/2505515.2505665. URL http://doi.acm.org/10.1145/2505515.2505665.

[19] Yoon Kim. Convolutional neural networks for sentence classification. In *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 1746–1751, 2014. URL http://aclweb.org/anthology/D/D14/D14-1181.pdf.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[21] Laurens van der Maaten and Geoffrey Hinton.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`.

[23] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 101–110, 2014. doi: 10.1145/2661829.2661935. URL `http://doi.acm.org/10.1145/2661829.2661935`.

[24] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pages 373–374, 2014. doi: 10.1145/2567948.2577348. URL `http://doi.acm.org/10.1145/2567948.2577348`.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.

[26] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL `http://arxiv.org/abs/1602.07261`.

[27] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015. URL `http://arxiv.org/abs/1511.04108`.

[28] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[29] Rui Yan, Yiping Song, and Hua Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 55–64, 2016. doi: 10.1145/2911451.2911542. URL `http://doi.acm.org/10.1145/2911451.2911542`.

[30] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Compu-*

tational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers, pages 643–648, 2014. URL http://aclweb.org/anthology/P/P14/P14-2105.pdf.

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