Degree project, 15 credits
for the degree of Bachelor of Science with a major in Computer Science
Spring Semester 2019
Faculty of Natural Sciences

A natural language processing solution to probable Alzheimer’s disease detection in conversation transcripts

Federica Comuni
Author
Federica Comuni

Title
A natural language processing solution to probable Alzheimer’s disease detection in conversation transcripts

Supervisor
Kamilla Klonowska
Birger Kleve, Sigma Connectivity

Examiner
Dawit Mengistu

Abstract
This study proposes an accuracy comparison of two of the best performing machine learning algorithms in natural language processing, the Bayesian Network and the Long Short-Term Memory (LSTM) Recurrent Neural Network, in detecting Alzheimer’s disease symptoms in conversation transcripts. Because of the current global rise of life expectancy, the number of seniors affected by Alzheimer’s disease worldwide is increasing each year. Early detection is important to ensure that affected seniors take measures to relieve symptoms when possible or prepare plans before further cognitive decline occurs. Literature shows that natural language processing can be a valid tool for early diagnosis of the disease. This study found that mild dementia and possible Alzheimer’s can be detected in conversation transcripts with promising results, and that the LSTM is particularly accurate in said detection, reaching an accuracy of 86.5% on the chosen dataset. The Bayesian Network classified with an accuracy of 72.1%. The study confirms the effectiveness of a natural language processing approach to detecting Alzheimer’s disease.

Keywords
Early Detection, Alzheimer’s Disease, Mild Cognitive Impairment, Bayesian Network, Long Short-Term Memory Recurrent Neural Network, Machine Learning, Natural Language Processing
6.3 LSTM ........................................................................................................................................... 34
7 Discussion........................................................................................................................................... 35
  7.1 Sustainability and ethical issues ................................................................................................. 35
8 Conclusion.......................................................................................................................................... 37
  8.1 Future work ................................................................................................................................... 37
9 References .......................................................................................................................................... 39
Appendices ........................................................................................................................................... 45
  Appendix 1: The Pitt Corpus and the Cookie Theft picture ............................................................... 45
    The Cookie Theft picture .................................................................................................................. 45
    Pitt Corpus tags relevant to this study ............................................................................................ 46
  Appendix 2: Natural language processing .......................................................................................... 47
    What is natural language processing? .............................................................................................. 47
  Appendix 3: Source code .................................................................................................................... 48
    Bayesian Network ............................................................................................................................ 48
    Baseline Neural Network .................................................................................................................. 51
    LSTM ................................................................................................................................................ 54
1 Introduction

This study proposes an accuracy comparison of two of the best performing machine learning algorithms [1] [2] in natural language processing, the Bayesian Network and the Long Short-Term Memory (LSTM) Recurrent Neural Network, in detecting Alzheimer’s disease symptoms in transcripts from the Pitt Corpus [3].

1.1 Background

Alzheimer’s Dementia is the most common type of senile dementia [4]. It consists of symptoms that vary in nature and intensity depending on its severity. At its most severe stage, dementia entails cognitive impairment, behavioral issues, difficulty in remembering events and aphasia, i.e., deterioration of language functions [5]. Aphasia reflects a loss of both lexical, semantic and pragmatic language processing in symptoms such as decreased vocabulary, use of repetitions, go-ahead utterances [5] and periphrases, substitution or mispronunciation of words, and a tendency to digress from the topic [6].

Recently, a branch of linguistics and artificial intelligence called natural language processing has progressed at a fast pace [7]. The ability of natural language processing algorithms to extract information from text has been employed by researchers to detect symptoms of neurodegenerative diseases in text samples. Studies have also demonstrated the efficiency of combining natural language processing with machine learning to achieve more accurate results when classifying text through supervised learning. This study therefore investigates the use of some of the newest machine learning algorithms to detect symptoms of Alzheimer’s Disease in the Pitt Corpus, a collection of transcripts of conversations with patients carried out by researchers from the Alzheimer and Related Dementias Study at the University of Pittsburgh School of Medicine [3].

The study is part of an on-going research that has flourished since the recent booming of machine learning and natural language processing. The two disciplines, however, have not always been coupled to detect Alzheimer’s disease. In 2016, Kim and Park [8] applied natural language processing without machine learning and analyzed the differences in lexical variety and morphology between early and late production of Iris Murdoch, diagnosed with Alzheimer’s disease, and Arthur Conan Doyle, about whom there are no known record of dementia. They found significant differences in the two authors’ writing and an evident language deterioration in Murdoch.

Machine learning has also been applied for the same purpose independently from natural language processing. In 2015 König et al. [9] applied signal processing techniques to voice recordings from Alzheimer’s disease patients and trained machine learning algorithms to distinguish Mild Cognitive Impairment (MCI) from Alzheimer’s disease, obtaining promising results. More studies have since combined the versatility of machine learning with the efficiency of natural language processing to detect Alzheimer’s disease in written and oral production.

1.2 Problem and Motivation

Alzheimer’s disease, also called Alzheimer’s Dementia (AD), is the most common type of senile dementia [4], affecting millions of people around the world. The Alzheimer’s Association reports:
An estimated 5.7 million Americans of all ages are living with Alzheimer’s dementia in 2018. This number includes an estimated 5.5 million people age 65 and older and approximately 200,000 individuals under age 65 who have younger-onset Alzheimer’s [4].

It also reports that the number of deaths from Alzheimer’s disease has increased by 123% from 2000 to 2015. Currently there is no known cure for Alzheimer’s disease, although pharmaceutical research has shown that symptoms can be relieved to some extent [4]. Life expectancy of the affected is significantly reduced; duration of survival is from 5 to 8 years after diagnosis [5]. The Alzheimer’s association stresses that early detection is crucial to allow the affected and their caregivers to prepare plans before further cognitive decline occurs, and to slow it down when possible [4].

Alzheimer’s disease becomes a significative problem as the world population gets older. The World Population Ageing Highlights issued by the United Nations in 2017 [10] reported almost a billion people aged 60 years or over, more than twice the number of 1980. That number is expected to double again by 2050 as a result of the global decrease of fertility rate and improvement of living conditions [10]. At the same time, technology use is increasing among seniors. The Pew Research Center of Internet and Technology reports that 40% of elderly people in the United States owned a smartphone in 2017, more than double the share from 2013 [11]. Companies around the world are shaping their goals and missions to adapt to the changes in society and to cater to the growing need of the aging population. One of these companies is Sigma Connectivity AB in Lund, a design and development house currently collaborating with Doro, a digital safety company providing easy-to-use smartphones and telecom services for seniors. The thesis serves to Sigma Connectivity as proof of concept, which will be seamlessly integrated into an ecosystem of smart household devices to help elderly people live a better and easier life.

While companies develop products to ensure seniors safety and independence, researchers are exploring the limitations of current Alzheimer’s disease detection methodologies and acknowledging the urgency to find more accurate tests. Orimaye et al. [12] report that current cognitive examinations, of which MMSE (Mini-Mental State Examination) is the most frequently used, suffer from limitations in distinguishing the subtypes of dementia (e.g. Vascular, or Parkinson’s) and rely on the ability and experience of the clinician performing the test. Weissenbacher et al. [13] claim that MMSE might fail to detect early stages of dementia, as it tends to abstract language from its psychological and social components. Since the early 2000s, the release on the Dementia TalkBank [14] of corpora of conversation transcripts, and the progress of natural language processing in text classification and text extraction have encouraged researchers to propose a natural language processing approach to detecting Alzheimer’s disease symptoms as a more accurate alternative to traditional cognitive tests. Such approaches have been deemed beneficial due to the fact that neurodegenerative disorders tend to first damage nerve cells that are responsible for speech and language processing [12].

1.3 Research questions

The study answers the following research questions:

**R1:** Can machine learning be used to detect symptoms of Alzheimer’s disease in text samples?

Research can only be successful and medically relevant if machine learning can detect symptoms of the disease in text samples.

**R2:** Which Alzheimer’s disease symptoms are most suited for selection for Alzheimer’s disease detection in conversation transcripts?
When considering the plethora of linguistic symptoms of Alzheimer’s disease and their verbal expression, and when investigating an efficient way to detect them with machine learning, some features might be more relevant than others.

R3: Which machine learning algorithm is best suited to use as a baseline when detecting Alzheimer’s disease in conversation transcripts?

R4: Which algorithm can be expected to perform better than the baseline, and why?

Supervised machine learning offers a wide variety of algorithms; the literature review explores the best options for this study.

1.4 Aim and purpose

The study aims to be a diagnostic tool for Alzheimer’s disease with a text analysis approach. Besides this diagnostic purpose, it also aims to compare the accuracy of two popular machine learning algorithms traditionally used for natural language processing. The algorithms are trained and tested on the cookie theft descriptions (see Appendix 1) from the Pitt Corpus [3]. The accuracy is measured as the percentage of true positives (texts from patients with AD classified as such), also called sensitivity, against false negatives (texts from patients with AD classified as healthy), true negatives (texts from healthy subjects classified as such) as well as false positives (texts from healthy subjects classified as AD).

The software implemented for this study can be applied to classify any other dataset in which the presence of Alzheimer’s disease is suspected. Since the algorithms do not diagnose dementia per se, but only the language deficits that derive from it, their application might also be useful in diagnosis of other neurodegenerative conditions with similar symptoms, such as primary progressive aphasia [15].

1.5 Limitations

1.5.1 Dataset

The Pitt Corpus is a relatively small collection, and within it, the cookie theft dataset is even smaller, containing 309 samples from patients with Dementia or MCI (Mild Cognitive Impairment) and 243 control samples. The relevance of this small size is considerable since, in statistic modeling, a part of the dataset needs to be used for training and the other for testing. The study has therefore been limited by the size of the dataset; nonetheless, research has shown that it is possible to achieve a decent level of accuracy with trained algorithms [2].

Limitations also stem from the fact that the dataset consists in transcriptions of verbal conversations, and not in texts written directly by patients; the transcriptions, in fact, cannot report any error such as misspellings, which have been proved relevant in Alzheimer’s detection based on written production [13]. With that said, researchers have transcribed the conversations with fidelity in reporting hesitation utterances (such as “ehm” or “hum”), word mispronunciations, incomplete and unintelligible words and trailing offs; these transcriptions can therefore be considered a faithful representation of the audio recordings.

1.5.2 Accuracy in previous work

The highest accuracy achieved by a machine learning algorithm in Alzheimer’s detection on the cookie theft dataset has been 91% from Karlekar, Niu and Bansal [2]. Other studies reached an
accuracy of 86.1% [12] and 87.5% [16]. Since a 91% accuracy means that, on 100 000 administered tests, 9000 would not be classified correctly, it is particularly important to integrate this test with other screening methods.
2 Method

This study has been conducted both through theoretical research and practical implementation of the machine learning algorithms, following pre-processing of the text samples. This section covers the dynamics of the former, while the latter is discussed in the Design and Creation section.

2.1 Literature Study

Because of the novelty of state-of-the-art natural language processing algorithms and the recent developments of the discipline [7], most of the literature review is based on peer-reviewed scholarship and articles from the Towards Data Science collection [17] instead of textbooks. The literature review aims at investigating previous work in the field, as well as building a strong foundation on natural language processing and on how it can be concretely applied to Alzheimer’s disease detection. The literature review also serves the purpose of finding the lexical, semantic and stylometric features most suitable for such detection, as well as giving an overview of the machine learning algorithms traditionally applied to this type of task.

2.1.1 Literature search

A selection of keywords was performed to yield pertinent results on Summon. The keywords “Natural language processing” and “Alzheimer’s” returned 11238 results, of which less than 10 were in fact related to both topics, and a few others were not directly related but still pertinent to both. The Summon research was integrated with an investigation on Kristianstad University’s Databases: the ACM Digital Library, PubMed and Wiley Online Library. This last research provided plenty of articles to explore the recent work in the field and form a solid background for this study.

The peer-reviewed scholarship articles can be subdivided into four categories, depending on their subject area:

1. The concrete applications of natural language processing, both with and without machine learning, to detecting Alzheimer’s disease in text samples
2. An analysis of interesting language features for Alzheimer’s disease detection, such as idea density or semantic similarity with Word2Vec [18], using machine learning
3. Purely medical articles on Alzheimer’s linguistic symptoms
4. The detection of Alzheimer’s disease applying machine learning to methods other than natural language processing, such as signal processing

The subcategories are sorted in descending order by degree of relevance to this study. From the first subcategory, five articles describe a similar work using the Pitt Corpus (alone or paired with others) and one using another dataset; the articles will be covered extensively in the Literature Review section.
3 Literature review

This section investigates the relevant literature and the previous work in the field to provide an answer to the research questions.

3.1 Related work

Rogers and Girolami [19] state that supervised machine learning is particularly suited to classify text, because of the laboriousness of the task of building a set of rules and models manually, and the large amount of data in the text with which classifiers can be trained. Alzheimer’s disease has been traditionally diagnosed with supervised machine learning in two ways: with natural language processing on text samples, and by analyzing Magnetic Resonance Imaging (MRI) scans. The former has often been preferred to the latter thanks to its efficiency compared to the analysis of MRI scans, which has shown an accuracy of 50% to 70% in case of Mild Cognitive Impairment and of 70% to 90% in Alzheimer’s disease in [20]. Natural language processing has sometimes been preferred to traditional clinical examinations as well because such examinations are expensive and often performed when the disease has already progressed to an advanced stage, thus becoming useless for early detection [13].

The efficacy of a natural language processing approach has been proven by numerous studies. Most notably, in November 2018 Beltrami et al. [21], by analyzing quantitively acoustic, lexical and syntactic features of spoken tests, demonstrated that natural language processing techniques can guarantee a higher accuracy in detection of language deterioration compared to traditional neuropsychological assessments, and therefore allow earlier diagnosis of cognitive decline. The study based its work on an article published in 2012 by Guinn and Habash [5] which discusses the features that should be considered relevant when performing language analysis of speakers with Alzheimer’s disease and when automatically distinguishing their production from the one of healthy subjects. The authors of the study found that specific features, such as go-ahead utterances, fluency, and paraphrasing, are distinctive of the linguistic production of patients with Alzheimer’s. Since then, numerous studies have chosen natural language processing to diagnose dementia, a summary of the studies is included below.

Natural language processing has progressed consistently in recent years [7]. For this reason, most of the previous work on the detection of Alzheimer’s disease in text samples has been conducted in the past 3 or 4 years. Due to the lack of publicly available datasets of writings or conversation transcripts from patients with Alzheimer’s disease, many studies have conducted their work on the Dementia TalkBank [14], and especially on its most comprehensive dataset in English, the Pitt Corpus [3]. A few other studies have either analyzed independently-collected datasets [13] [21] (not publicly available) or transcripts from the Carolinas Conversations Corpus [22], a collection of conversations with dementia patients about health and cognitive functions [23], sometimes together with the Pitt Corpus [24]. Nonetheless, results were very promising throughout the examined articles about the possibilities of detecting different types of dementia on the available datasets.

In 2014, Rudzicz et al. [24] proposed a machine learning approach to identifying trouble indicating behaviors (shortened in TIBs) with the purpose of creating a smart dialog software that could help AD seniors resume interrupted communication, while also identifying confusion in the speaker. TIBs include requests for repetition, comments such as “I can’t remember”, corrections of semantic inaccuracy, and others. The researchers extracted over 200 lexical, syntactic and acoustic features, of which only 5 have been listed, including number of words per minute and percentage of strong neutral words. The researchers then trained two machine learning algorithms (Naïve Bayes and Support Vector Machine) to identify TIBs in the Dementia TalkBank and Carolinas Conversations Corpus. Naïve Bayes yielded the highest accuracy of 79.5% on the Carolinas Conversations Corpus.
Even though Rudzicz et al. did not attempt to detect Alzheimer’s disease per se, the relevance of trouble indicating behavior and confusion to this type of dementia [6], as well as the detection of features like repetitions and corrections, made the article an important reference for this study.

In 2016, Orimaye et al. [16] proposed a comparison between a Neural Network Language Model (NNLM), A Deep Neural Network Language Model (DNLM) and a Deep-Deep Neural Network Language Model (DDNNLM) approach to detecting Mild Cognitive Impairment in verbal utterances on the Pitt Corpus. A language model, in natural language processing, is the probability of distribution of sequences of words. The authors considered two features for classification: the vocabulary space of 6-grams (i.e. groups of 6 words in immediate sequential order) and the space of 1-skip-trigrams (i.e. groups of 3 words in non-immediate sequential order, skipping 1 word in between). The DDNNLM performed better than the other two in all tests, yielding a lower percentage error (12.5%) and perplexity (1.6), which measures the level of “confusion” language models have in predicting words. While innovative and highly performant, this study proposes a strictly language model-focused approach, centered exclusively around grams. The addition of other features can provide a more in-depth approach to Mild Cognitive Impairment detection, as demonstrated by the following studies.

In 2016, Weissenbacher et al. [13] collected a set of descriptive writings from 201 healthy patients and patients with Alzheimer’s disease from the Arizona Alzheimer’s Disease Center. The writings describe the elements portrayed in a given picture, similarly to the Pitt Corpus, with the difference that the former are not conversation transcriptions, but genuine written samples. The authors then selected lexical, stylometric and semantic features, including:

- Ratios of adjectives, nouns, verbs and pronouns compared to the total number of tokens in the text. This feature was selected due to the hypothesis that AD positive patients use a smaller number of adjectives and pronouns, consequently to the vocabulary impoverishment in Alzheimer’s
- Type-token ratio, given by the number of unique lemmas in the text divided by total number of tokens in the text. This feature was also proposed due to the decrease of lexical variety in Alzheimer’s
- Functional words ratio, given by the number of functional words (i.e. prepositions, adverbs and pronouns like at, to, that, etc.) divided by the total tokens
- Character 5-grams: as in the previous study, the researchers considered n-grams (of characters in this case, of words in the other) of relevance to detecting the author’s style. The most frequent characters in both ill and healthy subjects were compared to classify test samples
- Idea density: heuristic that evaluates the amount of information conveyed in a text, following the hypothesis that AD positive patients tend to convey less information than healthy subjects. It is measured by calculating the ratio of adjectives, adverbs, verbs and conjunctions [13]
- Word2Vec Distance: a collection of words translated to vectors with the Word2Vec algorithm, where closer vectors represent words semantically closer. The texts were compared by similarity to a control text from a healthy subject, due to the tendency of Alzheimer’s Disease patients to digress from the topic

The authors used the selected features, as well as subject features like age and gender, to train different machine learning algorithms and then classify the test samples. The best performing algorithm was a Bayesian Network with an accuracy of 86.1%.

In 2017, Orimaye et al. [12] applied machine learning to detect Alzheimer’s disease in the Pitt Corpus. As in the study mentioned above, the authors selected lexical and semantic features, of which the most relevant are the following:
• Total number of coordinated sentences
• Total number of subordinated sentences
• Total and average number of predicates, i.e. elements of a sentence containing a verb and referring to the subject
• Total number of utterances
• Mean length of utterances
• Total number of functional words
• Total number of unique words, as the number of words that are never repeated within the text
• Total sentences in the text

These features were chosen as indicative of syntactic or lexical complexity, following the hypothesis that AD positive texts tend to be brief and simple. Further features are:

• Number of repetitions, as the occurrences in which the speaker repeated the same word twice or more times
• Number of revisions, as the instances where the subject, making a mistake, retraced it and corrected it
• Number of trailing offs indicators, i.e. instances in which the speaker left a sentence or word incomplete due to distraction or daydreaming
• Number of incomplete words, as utterances where the subject did not pronounce all syllables of a word
• Number of filler words, i.e. utterances that express hesitation or confusion, e.g. *hum*

The above features were picked as symptoms of confusion or misunderstanding of the given task, and therefore indicative of the presence of Alzheimer’s disease in the author. Bigrams and trigrams were also used. The researchers then applied these features to train the machine learning algorithms, together with the age of the subject. The listed features proved relevant in diagnosing Alzheimer’s disease in the Pitt Corpus text samples.

In 2017 Fraser, Fors and Kokkinakis [25] proposed a combined supervised and unsupervised machine learning analysis to speech samples from patients with Mild Cognitive Impairment, in English and Swedish. The datasets used in the study were the Pitt Corpus and two Swedish corpora, the Gothenburg and Karolinska datasets, both including descriptions of the cookie theft picture. As in Rudzicz [24], the authors considered timestamps of the audio recordings to select relevant features, which include:

• N+V density, as the total number of nouns and verbs divided by the total number of words
• N+V efficiency, as the total number of nouns and verbs divided by the total time of the narration in seconds
• Idea density

and other features found through clustering. The researchers concluded that adding a Mild Cognitive Impairment-positive dataset in a second language yields better results in detecting Mild Cognitive Impairment compared to adding a control dataset in the same language.

In 2018 Karlekar et al. [2] applied a Convolutional Neural Network, a Long Short-Term Memory Recurrent Neural Network and a combination of the two to the Pitt Corpus for classification. The third algorithm classified test samples with an accuracy of 91.1% on Parts-Of-Speech tagged data (using the Parts-Of-Speech tags provided in the Pitt Corpus). The simple LSTM model performed with an accuracy of 83.7%. The error margin was mostly due to the number of false positives, rather than false negatives. The researchers then applied unsupervised machine learning to cluster
text samples by feature. The automated clustering confirmed previous studies by showing a higher occurrence of the following attributes in AD-positive samples:

- Short or monosyllabic sentences
- Requests for clarification
- Starting with interjections, e.g. oh, well or right

The reviewed studies have therefore provided insights on the techniques to employ and features to extract for an efficient dementia diagnosis.
4 Design and Creation

This section describes the design and functionalities of the software components: the text pre-processing, the Bayesian Network, the baseline Neural Network, and the LSTM.

4.1 Design and theory

4.1.1 Bayesian Network

The Bayesian Network is a type of directed acyclic graph describing causation between nodes (i.e. random variables) through edges. The conditional dependence between nodes can be inferred through Bayesian inference [26]. The model for this study can be explained as follows:

Given two classes \( d \) (dementia) and \( c \) (control), the probability of a sample belonging to the dementia class is defined as \( P(d) \), while the probability of a sample belonging to the control class is defined as \( P(c) \). In this example, the dementia-positive texts are 309 out of 552, therefore \( P(d) \) is 0.56 and \( P(c) \) is 0.44. Given a feature \( id \) (idea density), the probability of a given value for \( id \) to belong to the dementia class can be calculated according to the Gaussian equation, where \( \mu \) is the mean of the values for the features and \( \sigma^2 \) is the features’ variance:

\[
P(id_x|d) = \frac{1}{\sqrt{2\pi \sigma_{id,d}^2}} e^{-\frac{(id_x-\mu_{id,d})^2}{2 \sigma_{id,d}^2}}
\]

The same probability can then be calculated for all features in both classes.

The probability of a given sample \( x \) belonging to the dementia class can then be computed by multiplying all the probabilities for its features, in this case \( id \) for idea density and \( a \) for age:

\[
P(x|d) = p(id_x|d) p(a_x|d)
\]

The model then calculates the probability of sample \( x \) belonging to the control class, and combines the two probabilities according to Bayes’ theorem:

\[
P(d|x) = \frac{P(x|d)P(d)}{P(x|d)P(d) + P(x|c)P(c)}
\]

The Bayesian Network follows the above process to calculate the probability that a given sample belongs to a class or the other.
4.1.2 Neural Network

A Neural Network is a computational model, vaguely inspired by biological neural networks, that represents a composite function:

\[ f(x) \rightarrow y \]

Where \( f(x) \) is a composition of other functions, which are in turn composed of other functions.

In supervised learning, a neural network can learn by being supplied with labeled data. The learning process implies that the neural network automatically adjusts its components to prioritize some characteristics of the input over others, in order to build the correct output [27]. The learning process also allows the network to generalize and to correctly classify a new, unlabeled dataset.

One of the main components of a neural network are the neurons (also called nodes). Each neuron represents the composition of an input function and an activation function, where the former is a weighted sum of all incoming data from the previous neurons, and the latter is a function that restricts the output to a certain range. The purpose of activation functions is to establish a threshold after which the neuron should fire, i.e. send its output to the following neurons. Machine learning uses a variety of activation functions; the ones used in this study are the Rectified Linear Unit (also called ReLU) and the Hyperbolic Tangent (also called tanh). The ReLU function is described by the following formula:

\[ R(x) = \max(0,x) \]

Where \( R(x) \) is 0 when \( x \) is less than 0, and \( R(x) \) is equal to \( x \) when \( x \) is greater than or equal to 0. **Figure 1** shows the graph of the ReLU function.

![Figure 1 - Graph of the Rectified Linear Unit function](image)

The tanh function is instead described by the following formula:

\[ \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

Where \( \tanh(x) \) assumes values between -1 and 1 for any value of \( x \): the larger \( x \) is, the closer \( \tanh(x) \) will be to 1; the smaller \( x \) is, the closer \( \tanh(x) \) will be to -1. **Figure 2** shows the graph of the tanh function.
Neurons are disposed into layers interconnected with each other, where each node receives input from all the nodes in the previous layer. The layers are typically called dense or fully connected layers, and a neural network can include an input layer, an output layer, and zero or more hidden, i.e. internal, layers. **Figure 3** depicts the architecture of a deep neural network, i.e. a network with two or more hidden layers.

**Deep neural network**

The input of neuron $n$ on layer $l_i$ is described as the sum of the product of the output of all neurons in the previous layer (layer $l_{i-1}$) and the weight of the connections between $n$ and the neurons on $l_{i-1}$. By calibrating the weights on the connections between nodes, the network can adjust the output of the function, so it matches the desired output specified through the labels. The network can learn by calculating how much its predictions differ from the desired output; this quantity is computed through a function called cost function, or error. In short, the network can improve its accuracy by iterating over the training dataset and adjusting its weights in relation to the cost function, in a process called backpropagation. The weights are adjusted proportionally to a
quantity called learning rate, usually a value between 0 and 1, which is multiplied to the estimated error rate. If the learning rate is set to 0.2, for example, the weights are updated at each iteration by 20% of the estimated error. The larger the learning rate, the faster the model will train, but with the possibility of missing the optimal value for the weights and therefore the best accuracy.

The number of layers and neurons per layer are hyperparameters, i.e. values set by the programmer to tune the neural network to its best performance. The activation function and the learning rate are also hyperparameters.

A neural network can learn thanks to algorithms that minimize the error rate by adjusting the weights [28]. One of the most popular of these algorithms is the gradient descent, which allows the model to learn the gradient for each parameter with respect to the error using backpropagation: for a given weight, the gradient descent uses the chain rule to calculate the derivative of the cost function for that weight, which represents the slope of the function, and adjusts the weight to reach a local minimum or the global minimum, where further tweaks produce no changes. The local or global minimum are also called convergence. When the error reaches its convergence, the weights have been optimized and the model should predict with its best accuracy.

4.1.3 LSTM

The LSTM is a special kind of Recurrent Neural Network. Recurrent Neural Networks are dense networks, with the difference that they allow information to persist [29], by refeeding output data to the nodes (see figure 4).

![Recurrent Neural Network and Feed-Forward Neural Network](image)

*Figure 4 - Structure of an RNN and a regular NN. The figure shows the “loops” that allow refeeding of the output data to the nodes*

The structure of Recurrent Neural Networks allows them to learn from previous inputs because the network shares the same weights with as many layers as chosen for a certain timestep. Even though this possibility to learn from past information has made Recurrent Neural Networks particularly efficient for language-related tasks such as prediction, translation or captioning [30], their architecture makes them susceptible to the problem of the vanishing gradient [29]: since the input of a neuron is not only given by the output of neurons on the immediately previous layer,
but also by the output of neurons on other previous layers, the chain of multiplications of the weights with multiple small values (i.e. values < 1), given by the chain rule of the backpropagation, gradually decreases the value of the weights and makes them vanish. The vanishing gradient problem also prevents the Recurrent Neural Network from handling long-term dependencies, i.e. from learning from information that traces back many layers [30].

Long Short-Term Memory networks (LSTMs) can prevent the vanishing gradient problem and learn from long-term dependencies, and they do so by replacing single network layers with more complex modules of four layers. Figure 5 illustrates schematically the four layers in each module, and figure 6 depicts the modules of a regular Recurrent Neural Network in comparison.

The LSTM modules contain three gates, consisting of a sigmoid layer combined with an operation of pointwise multiplication (represented by the pink circle with an X in figure 5). These three gates serve the purpose of deciding whether information should be forgotten, i.e. discarded, or passed on to the following layers. The tanh layer serves the purpose of choosing which new information should be added to the output of the module. Thanks to this complex structure and to the combination of new information with the persisting data, LSTMs can learn from long-term dependencies without incurring in the vanishing gradient problem.
4.2 Implementation

4.2.1 Text pre-processing

The conversations from the Pitt Corpus were transcribed in the CHILD format [31], which includes a header with metadata about the audio recording and the subject (age, gender, eventual diagnosis) as well as the utterances from both subject and interviewer, with the respective morphological and grammatical tags (see figure 11 in the Results section). Consequently, the conversation files needed to be pre-processed in order to extract the interesting features. The extracted features were then written to .csv (comma-separated values) files.

A few rounds of text pre-processing were performed. First, relevant information such as age, gender and diagnosis were extracted from the header. The header and all transcriptions from the interviewer were then discarded. Secondly, the remaining text was separated into three contiguous parts: the first with the whole patient’s transcriptions, the second with its whole morphological tagging, and the third with its whole grammatical tagging. Thirdly, relevant features such as repetitions, filler words, etc. (as it will be discussed in the Literature Review Results section) were extracted from the first subsection of the files using the tags provided by the Pitt Corpus. Thanks to the morphological tags, the text was then cleaned of all functional words and only nouns, verbs, adjectives, and adverbs were kept in order to count the ratio of these parts of speech. Finally, the text was completely cleaned of all tags to be prepared for use with Word2Vec and calculation of idea density.

Extraction of the number of subordinates and coordinates was carried out by matching the sentences against a dictionary of coordinating and subordinating conjunctions and counting their occurrence. The type-token and the adjectives and adverbs ratios were found thanks to the morphological tags provided by the corpus. The idea density was extracted using CPIDR 3 [32] (Computerized Propositional Idea Density Rater, version 3), a software released in 2007 by researchers at the University of Georgia that automatically tags text into parts of speech and calculates the ratio between the occurrence of adverbs, adjectives, verbs and pronouns and the total number of words in the text. Finally, the Word2Vec distance was calculated by implementing a Python application with Gensim [33], an open-source Python library targeted specifically to natural language processing, and Pandas [34], an open-source library providing highly performant tools and data structures for data analysis in Python. The Word2Vec distance was calculated using a pretrained model, which was found by analysing the ten most similar words to “cat”. The Google News model [35] returned words not always strictly pertinent to “cat”, such as “horse”, “dog” and “lady”, while the Fasttext Wikipedia model [36] returned more semantically related words like “kitten”, “feline” and “super-cat”, and was therefore chosen for the application. The cosine similarity between each word of the text samples and each word of a purposely produced control text was then computed, and the mean of the similarities of the words was used as a feature for the classification.

Age was also extracted from each text sample’s header and used as a feature. Since not all samples included the age, the missing data were replaced by the mean of all ages for the same class. All features were then stored on a .csv file including the name of all features and the diagnosis label for each sample (“Y” for dementia-positive texts, and “N” for healthy texts).

4.2.2 Splitting of the dataset

In machine learning, classification is based on the statistical analysis of features occurring in the training set, and on the probability inference that samples belong to a class depending on their features. The model will predict incorrectly if the distribution of the features varies substantially between the training set and the test set. Choosing the correct ratio for training, validation and test set is therefore fundamental to make the latter sets statistically representative of the whole set.
According to the central limit theorem, for samples with independent features – as it is assumed for this study - the larger the sample size is, the more similar its mean and standard deviation will be to the whole population’s mean and standard deviation [38] [39]. It can be inferred that, in supervised learning, the training set should be large enough for the model to learn effectively from its features, while the validation and test sets should be large enough to be statistically representative of the whole set. It was therefore decided that, for a small sample such as the one proposed in this study, a higher ratio of validation set and test set to training set, in the range traditionally proposed for supervised learning [40], was necessary for the model to predict accurately. A ratio of 64% training set to 16% validation set and 20% test set was therefore chosen.

5-fold cross-validation was used to avoid overfitting the model for the Bayesian Network and the baseline Neural Network [41]: a preliminary training set was first randomly shuffled and then, for five times, split into five equal parts: one fifth, kept as validation set, and four fifths, kept as training set. For each of the five times, the model was then trained on the training set. In the end, the model was evaluated on the training set, the validation sets, and the test set. This process prevented the algorithm from fitting too strictly to the former and perform less accurately on the latter. The preliminary training set was shuffled using the same random seed across the three models, to give comparable results when iterating on network design.

### 4.2.3 Bayesian Network

The Bayesian Network was implemented in Python using Scikit-learn [42], a collection of tools for data mining and machine learning. It was assumed that features were independent of one another and not linked by causal probabilistic relation, and a Naïve Bayes classifier was therefore chosen for the model. Even though Bernoulli and Multinomial are the two most popular Naïve Bayes classifiers for text classification [43], their suitability to classify, respectively, binary data and word frequency, made them inappropriate for this study, which uses a combination of discrete values (e.g. number of sentences) and continuous values (e.g. parts-of-speech ratios, idea density) with a strong prevalence of the latter. Furthermore, plotting the frequency distribution of the features showed that most of the features occurred in a normal distribution, which is best described by the Gaussian Naïve Bayes classifier [43] (see figure 7). The Gaussian Naïve Bayes classifier was therefore chosen for the model. Figure 8 shows the Naïve Bayes architecture of the model: the features are linked to dementia, meaning that there is a causal probabilistic relationship between them and the disease, without being related to each other, i.e. no causal relationship is assumed between the features.

The application was implemented to read the features values from the first partition (80% of the total dataset) and store them in a list. It then split the list into two subsets: one set containing the features values for each text sample, and the other containing the label for each sample in the same order as the former set. 5-fold cross-validation was then performed on the sets by shuffling them randomly but keeping the same respective order. The model then evaluated its classification accuracy on train, validation and test set, as well as the respective specificity and sensitivity rates.
Figure 7 - Frequency distribution of idea density, age, and mean utterance length: the frequencies follow a normal (or Gaussian) distribution

Figure 8 - Architecture of the Bayesian Network: the arrows indicate a causal probabilistic relation where the element at the origin of the arrow causes the element at the destination of the arrow. Please note: the number of features has been reduced for layout purposes
4.2.4 Baseline Neural Network

A 4-layer artificial neural network was implemented in Python with Keras [44] to provide a comparison with both tested algorithms and to use as a baseline for the LSTM.

In a process like the one followed for the Bayesian Network, the dataset was split into a set with the raw data for each text sample and a set with the corresponding labels. 5-fold cross-validation was then performed to ensure a correct fit of the model to the validation and test sets. The data were scaled using Scikit StandardScaler library [45] before feeding into the Neural Network, i.e. the feature values were normalized by removing the mean and scaling each sample to unit variance. Scaling was deemed necessary for the accuracy of the model because of the difference in value range between absolute features (e.g., the absolute number of utterances or sentences) and ratioed features (e.g. type-token ratio) [45]. The model was then trained on the training set and the accuracy, specificity, and sensitivity were evaluated on all sets.

The architecture of the model is depicted in figure 9. The Neural Network presents one input layer, two hidden layers, and one output layer. The number of layers and of nodes per layer was determined by using the heuristic approach of evaluating the architecture with the best accuracy, as suggested in [46]. The input layer has 22 neurons, i.e. one neuron for each feature, while the two hidden layers have 12 and 8 neurons. The output layer has one node because the model performs a binary classification [47]. The layers were all dense layers [48]. The rectified linear activation function was chosen for the layers because literature studies recommend it as the most suited activation function for regular neural networks [49]. A dropout rate of 0.2 was added after the first three layers to prevent overfitting the model [48]. Early stopping was used to automatically stop training of the model once the best accuracy for the validation set was reached (i.e. the validation accuracy reached a peak and then started decreasing due to overfitting). The number of epochs was set to 100. The adaptive moment estimation (commonly known as “Adam”) optimizer was chosen as optimizer for the model because of its proven efficacy over other optimizers in deep learning models [50]. The Adam optimizer works by choosing an individual learning rate for every single parameter, and by updating it at every training iteration. It is therefore an optimized variant of the gradient descent algorithm mentioned in the Design sections.

Figure 9 - Architecture of the Neural Network: the 22-node input layer is on the left-hand side while the output layer is on the right-hand side. Data flows from left to right
4.2.5 LSTM

The LSTM model was considerably more complex than the previous two algorithms: it was built, in fact, on three separate architectures that were then merged together (shown in figure 10). The three architectures were implemented due to the choice of using not only the extracted features, but also the text itself to classify the samples, thus using the LSTM as a language model [51]. Merging features and text is a solution that combines the methodology proposed by Orimaye et al. [12] and Weissenbacher et al. [13] with the one proposed by Karlekar et al. [2].

In order to use the text to train the model, the utterances were converted to integers using word embeddings according to the method proposed in [52]. Words of the text samples were matched to their vectors on a 300-dimension plane, based on their occurrence on a collection of Wikipedia 2017, UMBC web-based corpus and statmt.org news dataset [53]. This representation of words through vectors is traditionally called *word embeddings* in natural language processing. The reference to each word’s vector was then stored in a dictionary, and a matrix was created with the indexes of each word in this dictionary. A sequence length, equal to the longest text sample (197 words), was chosen and shorter samples were padded with 0s. The result of this process was a matrix that was used as input of the embedding layer of the LSTM. The LSTM model was then built in three layers as follows: the previously mentioned embedding layer served as input layer for the model, and then two bidirectional LSTM layers with 256 and 128 nodes each and a dropout of 0.2 were stacked to build the LSTM architecture. The bidirectional model was chosen because of its proven efficacy over the unidirectional LSTM with regression and classification tasks [54]. The second architecture was built out of a dense layer, with the raw data from the extracted features as input and 24 neurons. The outputs from the first two architectures were then merged in a 9-layer neural network with 8 dense layers and one batch normalization layer, which normalized the activation of the previous layers [55]. The nodes in the layers were in descending number, with one 84-node layer, three 64-node layers, one 32-node layer, two 16-node layers and a 1-node output layer. The hyperbolic tangent was chosen as activation function for all layers.

![Figure 10 - Architecture of the LSTM. The bidirectional LSTM layer is on the left-hand side while the dense input layer is on the right-hand side. On top is the neural network, which takes the embedding and the features as input. Data flows from the bottom to the top. Please note: the diagram is not an exact representation of the software architecture; the layers and nodes have been simplified for layout purposes.](image-url)
because of its proven efficiency for Recurrent Neural Networks [49]. A dropout rate starting from 0.5 and up to 0.2 was added after each layer to prevent overfitting. The hyperparameters of the architecture were tuned through experimentation. The inputs from the first two architectures and the output from this last architecture were then fed to the model, which was compiled with Adam optimizer. The model was then trained with early stopping and the accuracy, sensitivity and specificity were evaluated for all sets.
5 Results

5.1 Literature review results

The analysis of the articles summarized in the Literature Review section shows that each team of researchers devised a different strategy to detect Alzheimer’s Dementia or other diseases in the given dataset. Nevertheless, the studies presented some similarities, like the frequent use of Support Vector Machines (3 studies out of 6) and the choice of features, primarily ratio of Parts-Of-Speech elements (4 out of 6) word or character n-grams (3 out of 6) and idea density (2 out of 6). The review can therefore provide an answer to the first two research questions:

R1: Can machine learning be used to detect symptoms of Alzheimer’s disease in text samples?

Not only can machine learning be used to detect Alzheimer’s disease with a satisfying accuracy compared to the analysis of MRI scans [20] and without the onerous cost of an MRI scan machine [56], but it can also detect symptoms of lesser-degree neurodegenerative diseases, like Mild Cognitive Impairment, thus demonstrating the relevance of this approach to early diagnosis of cognitive diseases. In this sense it is important to highlight that, out of 6 studies, 5 were based on the Pitt Corpus (3 exclusively and 2 in addition to other datasets), which only includes diagnoses for Mild Cognitive Impairment and probable or possible AD, thus excluding any severe or even moderate AD samples.

R2: Which Alzheimer’s disease symptoms are most suited for selection for Alzheimer’s disease detection in conversation transcripts?

The researchers of the examined articles selected a variety of lexical, syntactic and semantic features, symptoms of the multifaceted language deficit of dementia. Overall, the features were selected on the following principles:

- Subjects with Alzheimer’s disease tend to write or say shorter and syntactically simpler sentences than the healthy population [12] [2], with a poorer lexical variety [13] [12] [24] [25]
- Subjects with Alzheimer’s tend to repeat themselves more often and to trace their errors [12]. They also tend to trail off, leave sentences or words incomplete, and to use filler words that express confusion or hesitation [12]
- Subjects with Alzheimer’s tend to request clarifications [24] [2] and to start sentences with interjections [2]
- They also tend to convey less information per total number of words [13] [25] and to digress from the topic [13]

The listed symptoms are documented by literature [6] and relevant to the detection of the disease in conversation transcripts. To select the features for this study, it is also important to consider the grammatical and morphological tags provided by the Pitt Corpus. As shown in figure 11 the Pitt Corpus provides, for each sentence, tags for grammatical analysis (noun, verb, adjective, etc.) and for logical analysis (subject, predicate, etc.). It also provides timestamps for each sentence and tags for other features including word repetitions, word replacements, trailing offs, incomplete words, and others (see Appendix 1).
In light of the literature review and tags provided by the Pitt Corpus, the following features have been selected for the study:

- Total number of coordinated sentences [12]
- Total number of subordinated sentences [12]
- Total number of predicates [12]
- Total number of utterances [12] [13]
- Mean length of utterances [12]
- Total number of unique words [12]
- Total sentences [12]
- Ratio of repetitions, given by the total number of repetitions divided by the total number of words [12]
- Ratio of revisions with correction, i.e. instances where the subject has retraced and corrected a wrong word or sentence [12]
- Ratio of revisions with reformulation, i.e. instances where the subject has retraced and semantically reformulated a sentence [12]
- Ratio of unintelligible words, i.e. words that could not be discerned by the audio recording transcriber (proposed)
- Ratio of filler words, e.g. *hum* [12]
- Ratio of trailing offs [12]
- Ratio of incomplete words [12]
- Ratio of prolonged syllables (proposed)
- Ratio of pauses in between syllables (proposed)
- Ratio of pauses in between words (proposed)
- Ratio of overlaps, i.e. instances where the subject overlapped the interviewer (proposed)
- Ratio of adjectives and adverbs [13] [25] [24]
- Type-token ratio [13]
- Idea density [13] [25]
- Word2Vec distance [13]
The features are the result of the selection of those presented by two of the most relevant works to this study [12] [13] and those made easily available by the Pitt Corpus tags. Age was also considered [13].

**R3: Which machine learning algorithm is best suited to use as a baseline when detecting Alzheimer’s disease in conversation transcripts?**

Three out of the six articles mentioned in the previous paragraph used a Support Vector Machine for classification. While the Support Vector Machine is indeed an efficient algorithm for Alzheimer’s detection and for text categorization in general [57], many other algorithms have been used throughout the years and it is impossible to pinpoint which of them is the absolute best [57]. Weissenbacher, in 2006 [1], measured the accuracy of a Bayesian Network in distinguishing anaphoric from non-anaphoric “it”, and proposed this algorithm as a valid tool for text classification in natural language processing. The validity of this type of network was then confirmed by the accuracy reached in detecting Alzheimer’s disease in the already mentioned study from ten years later [13]. In 2008 Ben-Gal [58] analyzed in depth the probabilistic mechanisms of Bayesian Networks and concluded that they represent an efficient tool for text mining and natural language processing because of their usefulness in structuring causal and probabilistic semantics. Because of these findings, the Bayesian Network was chosen as a baseline for this study.

**R4: Which algorithm can be expected to perform better than the baseline, and why?**

Convolutional and Recurrent Neural Networks have recently risen in popularity and are now considered the two state-of-the-art algorithms for natural language processing [59]. In 2017, Yin et al. [59] compared the performance of a Convolutional Neural Network and of the two most common forms of Recurrent Neural Network, the LSTM (Long Short-Term Memory) and the GRU (Gated Recurrent Unit) in tasks like sentiment classification, answer selection and Parts-Of-Speech tagging. In sentiment classification, the LSTM performed second best after the GRU with an accuracy of 84.5%. Overall, it was found that Recurrent Neural Networks tend to perform better than Convolutional Neural Networks when the text classification requires a comprehension of global or long-span semantics, as it is the case for detection of Alzheimer’s disease [59]. In particular, LSTMs are able to store information thanks to long-term dependencies and to select the features to store or remove from memory [30] [51], and this makes them particularly popular in natural language processing tasks. The LSTM Recurrent Neural Network was therefore chosen as algorithm to compare to the baseline.

### 5.2 Empirical results

#### 5.2.1 Accuracy

As it was hypothesized in the last research question, the LSTM model performed better than the Bayesian Network, and it also performed better than the baseline Neural Network. The results are summarized in table 1, 2 and 3. For all algorithms, the model classified with similar accuracy on the validation set and the test set. The test set was classified with a 72.1%, 81.1% and 86.5% accuracy in the Bayesian Network, baseline Neural Network and LSTM respectively. The accuracy reached by the LSTM on the test set (86.5%) sets a new benchmark accuracy for the LSTM on the chosen dataset.
**Table 1 - Results of the Bayesian Network**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Precision (specificity)</th>
<th>Recall (sensitivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>72.8%</td>
<td>81.6%</td>
<td>65.6%</td>
</tr>
<tr>
<td>Validation</td>
<td>72.7%</td>
<td>86.1%</td>
<td>63.5%</td>
</tr>
<tr>
<td>Test</td>
<td>72.1%</td>
<td>69.4%</td>
<td>74.2%</td>
</tr>
</tbody>
</table>

**Table 2 - Results of the baseline Neural Network**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Precision (specificity)</th>
<th>Recall (sensitivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>83.8%</td>
<td>82.3%</td>
<td>85.1%</td>
</tr>
<tr>
<td>Validation</td>
<td>81.8%</td>
<td>77.8%</td>
<td>84.6%</td>
</tr>
<tr>
<td>Test</td>
<td>81.1%</td>
<td>75.5%</td>
<td>85.5%</td>
</tr>
</tbody>
</table>

**Table 3 - Results of the LSTM**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Precision (specificity)</th>
<th>Recall (sensitivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>91.5%</td>
<td>96.8%</td>
<td>87.2%</td>
</tr>
<tr>
<td>Validation</td>
<td>86.4%</td>
<td>88.9%</td>
<td>84.6%</td>
</tr>
<tr>
<td>Test</td>
<td>86.5%</td>
<td>87.8%</td>
<td>85.5%</td>
</tr>
</tbody>
</table>
Table 4 summarizes the results of all models on the test set.

Table 4 - Specificity, sensitivity and accuracy for the test set

The rates of specificity, also called precision, and sensitivity, also called recall, varied extensively throughout the three models. The Bayesian Network’s sensitivity was almost 7% higher than the specificity; similarly, in the baseline neural network, the sensitivity was 13% higher. An opposite trend occurred in the LSTM, which classified with almost 3% more specificity than sensitivity.

The dropout added after each layer of the neural networks was effective in preventing the model from overfitting: the training set reached a mean accuracy across the three models of 82.7%, a rate only 2.9% higher than the validation set (mean accuracy 80.3%) and 3.5% higher than the test set (79.9%). On average, classification on the validation set was slightly more accurate (0.5%) than on the test set. Table 5 shows the mean specificity, sensitivity and accuracy rates across all models for the three sets.
Table 5 - Mean specificity, sensitivity and accuracy of the models on the training, validation and test sets

In order to have a deeper understanding of the performance of the three models, the accuracy of randomly classifying the sets must be calculated. In a model with two classes, the number of true positives ($TP$), true negatives ($TN$), false negatives ($FN$) and false positives ($FP$) can be calculated as follows [60]:

$$TP = pn +$$

$$TN = (1 - p)n -$$

$$FN = pn -$$

$$FP = (1 - p)n +$$

Where $n+$ is the number of dementia-positive texts (62 in the test set), $n-$ is the number of control texts (49 in the test set) and $p$ is the probability of a text belonging to a class (0.56 for dementia-positive texts, 0.44 for control texts). By plugging in the data for the test set, we find that $TP = 34.7$, $TN = 27.4$, $FN = 21.5$ and $FP = 27.3$. We can then measure the accuracy of the whole classifier by calculating its $F_1$ score, according to this formula [60]:

$$F_1 = \frac{TP + TN}{TP + TN + FP + FN}$$

We find that the accuracy of a random classifier is 50%. All three models therefore have abundantly exceeded the random accuracy.
5.2.2 Training efficiency

As mentioned in the Method section, 5-fold cross-validation was performed on the Bayesian Network and the baseline Neural Network to prevent overfitting and develop a model as versatile as possible. Figure 12, 13 and 14 show that, in the Neural Network, the model’s accuracy on the validation set increased across the training rounds and the model’s loss decreased, meaning that the model became capable of generalizing its predictions. The number of epochs varied from one round to the other and it never reached the hard-coded amount of 100 due to early stopping.

Figure 12 – NN accuracy and loss on the training and validation sets after the first round of training

Figure 13 - NN accuracy and loss on training and validation sets after the third round of training

Figure 14 - NN accuracy and loss on training and validation sets after the fifth round of training
Figure 15 shows the model’s performance on the LSTM: even in this case, the performance of the model on the validation set became closer to its performance on the training set throughout the epochs, stopping automatically after approximately 50 epochs.

5.2.3 Training speed

Recurrent Neural Networks are notorious for being slow to train [51], which was confirmed by the results of this study. Table 6 summarizes the time required to train all models, which were tested on a desktop with 32Gb of RAM and an Intel Core i7 7th generation processor on 100Kb of raw data for the extracted features, and around 350Mb of data for the embedded text. The LSTM software was particularly slow when it included the process of embedding the text, while it ran faster when the model was pretrained and the embedded text was loaded from a binary file.

Table 6 - Time required to train each model

<table>
<thead>
<tr>
<th>Model</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian Network</td>
<td>0.03 seconds</td>
</tr>
<tr>
<td>Neural Network</td>
<td>15 seconds</td>
</tr>
<tr>
<td>LSTM without embedding</td>
<td>5 minutes</td>
</tr>
<tr>
<td>LSTM with embedding</td>
<td>Almost 9 minutes</td>
</tr>
</tbody>
</table>

5.2.4 Features

The correlation between features was plotted to find which features strongly correlated to each other. As mentioned in the Literature Review Result section, the features that are characteristic of an elaborated and complex speech should not be positively correlated with the features that indicate confusion, hesitation or misunderstanding of the task. The results are shown in figure 16.
From the figure it can be observed that the number of utterances, sentences, predicates, unique words, subordinates and coordinates (all indices of complexity of speech) are strongly positively correlated to each other; they also negatively correlate with the type-token ratio, i.e. the ratio of unique lemmas to the total number of words in the text. It can be inferred that the longer the text, the smaller the proportional amount of unique words (it seems counterintuitive, however, that the absolute number of unique words negatively correlates with the type-token ratio). Age slightly positively correlates with dementia-positive features (e.g. hesitations, pauses between syllables) and slightly negatively correlates with dementia-negative features (e.g. length of utterances and subordinates). The number of trailing offs strongly positively correlates with the number of long syllables and unintelligible words, indicating that subjects who tended to leave sentences incomplete also tended to prolong syllables and to pronounce words not clearly. The length of the utterances negatively correlates with the number of utterances, predicates, sentences, unique words, subordinates and coordinates, meaning that subjects who tended to speak the most also tended to choose shorter words. As expected, the idea density positively correlates with dementia-negative features and negatively correlates with dementia-positive features. Overall, the features correlate as expected and it can be therefore inferred that they represent correctly the presence or the absence of Alzheimer’s disease in the author of the text.
6 Analysis

The accuracy rate reached by the algorithms in the study abundantly exceeds the random accuracy of 50% for binary classification on the chosen dataset. The results show that the hyperparameters, chosen through an experimental approach, have proven successful in efficiently configuring the models and prevent overfitting. Below is an evaluation of the performance of all algorithms.

6.1 Bayesian Network

The Bayesian Network classified with an accuracy of 72.1%. While this accuracy is considerably higher than the random accuracy, it is still far from the 86.5% rate reached by the LSTM. This might have been due to the fact that the model, based exclusively on the probabilistic relationship between the 22 features and the presence of dementia, was not complex enough to detect the small nuances of some of the texts, which only presented light symptoms (see Appendix 1). A manual inspection of the texts, in fact, revealed that a few samples were relatively hard to classify for a human because they did not present particularly strong features. The Bayesian Network classified with higher sensitivity than specificity on the test set; this performance makes this algorithm suited for screening tests because they require a higher sensitivity and therefore a lower chance of misclassifying positive subjects [61].

6.2 Baseline Neural Network

The Baseline Neural Network was created for the purpose of tracking progress and comparing its performance with the LSTM. It reached an accuracy of 81.1% and it still lacked the complexity to bring significant results in Alzheimer’s diagnosis on the Pitt corpus. That might be the reason why previous studies opted for more complex neural models for the same task [2] [16].

6.3 LSTM

The LSTM proved to be the best performing algorithm, in line with the findings exposed in the literature review, and its accuracy of 86.5% on the test set defines a new benchmark for LSTM performance on the Pitt corpus. The model presented a slightly higher (3%) specificity than sensitivity, but the high flexibility and possibility to tune the hyperparameters in neural models should make it feasible to increase the sensitivity. The high accuracy achieved by the model confirms the effectiveness of providing both the extracted features and the text as input for the algorithm. Feeding the text as input allows to use the LSTM as a language model, i.e. as a model that recognizes patterns in the text and finds relevant features automatically. The LSTM was also the most complex model, presenting over a million trainable parameters in its embedding layer. The only downside to the LSTM is that, due to its complexity, it requires an important amount of diversified data to be modelled effectively on the data’s nuances and features, and to classify with accuracy.
7 Discussion

The large-scale application of the findings of this study, either in industry or public institutions, should first address the concerns regarding the limitations outlined in the Introduction and in theEmpirical Results sections. As it was mentioned before, this study has been conducted on a particularly small dataset, with 552 samples and a total file size of around 3Mb. Due to its complexity, the LSTM requires to be trained on larger datasets, preferably in the order of several Gb, to better generalize its predictions and be able to classify any sample data more accurately [62].

Training of the LSTM has proved to be slow, especially when it includes text embedding; however, since the algorithm requires to be trained only once to be able to classify new samples, the training time should not affect the choice of adopting the best performing algorithm on large scale. Companies and public institutions should therefore opt for the most accurate algorithm, even if it requires more computational power and time for training compared to less accurate ones. The inference time should be taken into consideration when planning to use the software on machines with little computational power, such as embedded devices or mobile phones: if inference is too slow, the application might require to be implemented on the Cloud and accessed through APIs instead of being integrated into portable devices. Even in that case, however, inference should be fast enough to allow smooth communication between the device and the server. The algorithms should first be tuned to obtain the highest sensitivity as possible: correctly classifying true positives is particularly important in early screening tests, where the purpose is to diagnose as few false negatives as possible, even at the risk of increasing the number of false positives [61]. The algorithm might successively be tuned again to increase the specificity and used as a second test to correctly diagnose the healthy subjects.

The features used in this study have all been automatically extracted thanks to the implemented software and third-party applications; no features were manually computed. Automatic extraction is a requirement on large-scale applications, which cannot support manual features extraction.

7.1 Sustainability and ethical issues

This study is based on the use of healthcare data from patients with Alzheimer’s and is therefore susceptible to the ethical discussions concerning Big Data that have been recently brought to public attention. In 2014, Crawford, Miltner and Gray [63] analyzed in depth the ethical implications of Big Data and their impact on politics and society. The researchers argued that the current growing interest in Big Data might encourage companies to use it without concerns about privacy and confidentiality, and that, even when users agree to policies about treatment of their data, they might do so because they feel coerced by the system and because the only possible alternative is withdrawing completely from technology. An article from Towards Data Science from 2018 [64] explains that the use of Big Data should follow ethical principles, including confidentiality and privacy of customer’s data and transparency about the use of such data. All these concerns are relevant to this study, especially when considering that text samples come from patients with Alzheimer’s, who are not always capable of understanding the implications of participating to the data collection or to give consent about its use. The above-mentioned discussions become also particularly relevant when considering the study’s big-scale application in industry or healthcare: companies and public institutions should invest in the legal and informatic protection of customer’s data and ensure that, whenever the senior directly involved in the study cannot give consent to the treatment of the data, approval is sought from their loved ones. Text analysis is particularly versatile but also dangerous in this context: it is not a big step to go from detection of Alzheimer’s to detection of distress [24] and from there to gather more information about the customer’s family and financial situation, or other private matters that the subject might not want to disclose. Data security, confidentiality, and transparency are therefore of primary importance in this study. Users of the application should also be informed about the
meaning of the detection accuracy, i.e. that a high probability of the presence of Alzheimer’s should not be interpreted as certainty.

From a sociological point of view, being able to detect Alzheimer’s very early can be beneficial and encourage the senior to take measures to relieve the symptoms or to prepare before further cognitive decline occurs [4]. Since Alzheimer’s disease sometimes causes behavioral changes [5], early detection might help the loved ones understand the behavior of the affected seniors and encourage them to take measures to relieve their condition. Certainly, the study does not directly affect the course of the disease, but it can indirectly improve the health and living condition of the affected seniors and their loved ones.

The study does not have relevant implications in sustainability and the environment, besides, in minor part, those that apply to heavy computational tasks: if companies or public institutions choose to rely on in-site servers for training the models, they should do so in an energy-efficient way to minimize the impact on the environment. The same principle applies to Cloud Computing providers that offer services for machine learning and data analytics [65]. This aspect is relevant to this study because of the complexity and slow training speed of the LSTM.
8 Conclusion

The study shows promising results on the possibility of detecting probable Alzheimer’s symptoms in samples of text. The empirical results confirm the hypotheses formulated in the research questions: machine learning can be an effective tool in early diagnosis of Alzheimer’s disease and the state-of-the-art algorithm, the LSTM, has performed with better accuracy than the baseline algorithm, the Bayesian Network. The study has therefore provided a comparison of the performance of three machine learning algorithms popular in natural language processing, including the baseline neural network, among which the LSTM has classified the test set with the highest accuracy.

The findings can be summarized as follows, in relation to the research questions:

1. *Can machine learning be used to detect symptoms of Alzheimer’s disease in text samples?*
   
   Yes, the empirical results, as well as the literature study, have confirmed that Alzheimer’s disease symptoms can be detected in text samples thanks to machine learning, with higher accuracy than a random classifier.

2. *Which Alzheimer’s disease symptoms are most suited for selection for Alzheimer’s disease detection in conversation transcripts?*
   
   A variety of lexical and semantical features, coupled with the age of the subject, has proved successful for detecting Alzheimer’s in all three algorithms. For the LSTM, the extracted features have been combined with word embeddings to increase the complexity of the model and improve its accuracy.

3. *Which machine learning algorithm is best suited to use as a baseline when detecting Alzheimer’s disease in conversation transcripts?*
   
   The Bayesian Network was chosen as a baseline due to the extensive literature recommending it as an effective algorithm for text classification due to its capability of representing probabilistic relationships between independent variables. The Bayesian Network classified the test set with 72.1% accuracy.

4. *Which algorithm can be expected to perform better than the baseline, and why?*
   
   The LSTM was documented as being the state-of-the-art machine learning algorithm for natural language processing tasks thanks to its complex structure, its ability to store selected memories of information passed through its layers, and its ability to handle long-term dependencies. The LSTM outperformed the Bayesian Network with an accuracy of 86.5% on the test set.

8.1 Future work

Future work on the study could revolve around four main areas:

1. Improving the generalization of the algorithms by training them on larger datasets: as it was mentioned in the Discussion section, the LSTM will benefit from being trained on large datasets due to its complexity.

2. Increasing the number of extracted features: the models’ accuracy could be improved by adding new features to the ones proposed in this study, such as the timing of pronunciation of the sentences [25], or occurrences where the subjects have started the sentence with an interjection [2].
3. Improving the model by integrating the LSTM with a CNN, or replacing the LSTM with GRU: the approach of training on both the features and the embedded text can be applied to the best-performing algorithm in [2], i.e. a combination of a CNN and an LSTM, to get better results. To improve training speed, the LSTM might be replaced with a Gated Recurrent Unit (also called GRU), i.e. a less complex version of an LSTM that, according to Yin et al. [59], performs just as well or even better than LSTM in many natural language processing tasks.

4. Extract different features to detect diseases other than Alzheimer’s: Rudzicz et al. [24] have successfully employed machine learning algorithms to detect trouble-indicating behaviors on the Dementia TalkBank data; text classification algorithms might therefore be trained to detect the presence of distress in the speaker, or situations of emergency.

The field of language processing is continuously expanding [7] and being integrated into artificial intelligence in order to make machines interact more effectively with humans. This study’s findings in detecting Alzheimer’s Dementia in text samples can open possibilities for further studies in the early detection of diseases through text classification and natural language processing.
9 References


Appendices

Appendix 1: The Pitt Corpus and the Cookie Theft picture

The Cookie Theft picture

The cookie theft picture is a drawing depicting an indoor scene of a boy and a girl stealing cookies from a cookie jar, which is shown in figure 17. The picture was proposed by Becker and Boller [3] in 1994 to both healthy subjects and subjects with varying degrees of dementia to describe orally and report everything depicted. The conversations have been recorded and transcribed by undergraduate students.

![Cookie Theft Picture](image)

*Figure 17 - The standardized Cookie Theft picture (Goodglass and Kaplan 1983)*

The transcriptions are part of a broader, longitudinal research on Alzheimer’s disease conducted by the University of Pittsburgh School of Medicine. The dataset is part of the Pitt Corpus, a collection of transcriptions from different kinds of conversations and verbal tasks from patients with Dementia, which is itself part of the greater collection of datasets from the Dementia TalkBank [14]. The cookie theft descriptions have been extensively used in Alzheimer’s disease-related studies due to their relevance in identifying language deficits in patients with Alzheimer’s as well as aphasia, as demonstrated by [12] and [66].

The conversations collected in the dataset come from patients with a milder degree of Alzheimer’s disease symptoms (classified as possible or probable AD), and MCI (Mild Cognitive Impairment), as well as control (healthy) subjects, and do not include samples from patients with moderate or severe AD. The reason for this choice is that it is relatively pointless for diagnostics purposes to detect Alzheimer’s when it is already at its most severe stages.

The descriptions have been transcribed in the CHAT [31] format; the text has been lemmatized and morphologically and grammatically tagged.
Pitt Corpus tags relevant to this study

Table 7 lists the features used in this study that were extracted thanks to the tags provided by the Pitt Corpus.

Table 7 - Pitt Corpus tags: the column on the right reports the linguistic feature and the column on the left reports the corresponding tag

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Linguistic feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>Repetition</td>
</tr>
<tr>
<td>[//]</td>
<td>Revision with correction</td>
</tr>
<tr>
<td>[///]</td>
<td>Revision with reformulation</td>
</tr>
<tr>
<td>xxx</td>
<td>Unintelligible speech</td>
</tr>
<tr>
<td>&amp;</td>
<td>Filler word</td>
</tr>
<tr>
<td>()</td>
<td>Incompletion or omission</td>
</tr>
<tr>
<td>wo:rd</td>
<td>Lengthened syllable</td>
</tr>
<tr>
<td>wo^rd</td>
<td>Pause between syllables</td>
</tr>
<tr>
<td>## or ###</td>
<td>Pause between words</td>
</tr>
<tr>
<td>+…</td>
<td>Trailing off</td>
</tr>
<tr>
<td>+&lt;</td>
<td>Overlap</td>
</tr>
</tbody>
</table>
Appendix 2: Natural language processing

What is natural language processing?

Natural language processing is an IT discipline that combines linguistics and artificial intelligence to make applications “understand” and process human language. It includes fundamental operations, of which the following are relevant to this study:

1. Lemmatization: the process of reducing a word into its lemma, i.e. its dictionary form (e.g. reduction of is into be, or better into good);
2. Word embeddings: translation of words in a text to vectors on a multidimensional plane where the distance between vectors represents how semantically similar the corresponding words are (the smaller the distance, the most similar the words);
3. Parts-Of-Speech tagging: the process of marking up words as nouns, verbs, adjectives (i.e. parts of speech), and so on.
Appendix 3: Source code

This appendix reports the source code for the Bayesian Network and the baseline Neural Network.

Bayesian Network

Below is the source code of the Bayesian Network.

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import pandas as pd

def read_csv():
    
    Reads data from .csv file
    
    # reads data and stores it into dataframe
    df = pd.read_csv('train_set', sep=',')
    df_test = pd.read_csv('test_set', sep=',')
    split_dataframe(df, df_test)

def split_dataframe(df, df_test):
    
    Splits dataframe into x (containing the data)
    and y (containing the respective labels)
    :param df: whole train dataframe
    :param df_test: whole test dataframe
    
    # takes only dementia column (which are the labels, Y for dementia
    and N for control)
    # and converts to numbers: 1 for Y and 0 for N
    y = df['dementia'].apply(lambda x: 1 if x == 'Y' else 0)
    # drops columns with empty values (pauses and retracing_reform)
    # and drops dementia column because it is not needed
    x = df.drop(columns=['dementia', 'pauses', 'retracing_reform'])

    y_test = df_test['dementia'].apply(lambda x: 1 if x == 'Y' else 0)
    x_test = df_test.drop(columns=['dementia', 'pauses',
                                'retracing_reform'])
    do_kfold_validation(x, y, x_test, y_test)
```
def do_kfold_validation(x, y, x_test, y_test):
    """
    Performs kfold cross validation to avoid overfitting
    :param x: train set with data, before splitting into train and validation sets
    :param y: train set with labels, before splitting into train and validation sets
    :param x_test: test set with data
    :param y_test: test set with labels
    """
    # initializes kfold with 5 folds, including shuffling, using 9 as seed for the shuffling
    kfold = KFold(n_splits=5, random_state=9, shuffle=True)

    # uses a Gaussian Naive Bayes classifier
    model = GaussianNB()

    # splits datasets into folds and trains the model
    for train_index, val_index in kfold.split(x):
        x_train, x_val = x.loc[train_index], x.loc[val_index]
        y_train, y_val = y.loc[train_index], y.loc[val_index]
        # training of the model
        model.fit(x_train, y_train)
        evaluate_accuracy(model, x_train, x_val, x_test, y_train, y_val, y_test)
        plot_correlation(x)

def evaluate_accuracy(model, x_train, x_val, x_test, y_train, y_val, y_test):
    """
    Evaluates accuracy on train set and validation set
    :param model: trained model
    :param x_train: train set with data
    :param x_val: validation set with data
    :param x_test: test set with data
    :param y_train: train set with labels
    :param y_val: validation set with labels
    :param y_test: test set with labels
    """
    # evaluation on train set
    score = model.score(x_train, y_train)
    print("{0:<35s} {1:6.3f}%".format('Accuracy on train set:', score * 100))

    # evaluation on validation set
    score = model.score(x_val, y_val)
    print("{0:<35s} {1:6.3f}%".format('Accuracy on validation set:', score * 100))

    # evaluation on test set
    score = model.score(x_test, y_test)
    print("{0:<35s} {1:6.3f}%".format('Accuracy on test set:', score * 100))

    # stores predicted labels for the train set
    # in an array
    y_pred = model.predict(x_train)
    evaluate_sen_spec(y_train, y_pred, 'train')
# stores predicted labels for the validation set
# in an array
y_pred = model.predict(x_val)
evaluate_sen_spec(y_val, y_pred, 'validation')

# stores predicted labels for the test set
# in an array
y_pred = model.predict(x_test)
evaluate_sen_spec(y_test, y_pred, 'test')

def evaluate_sen_spec(y_true, y_pred, set):
    
    #param y_true: dataframe with data
    #param y_pred: dataframe with predicted labels for data
    #param set: type of set (train, val or test)
    
    # gets number of true negatives (tn), false positives (fp),
    # false negatives (fn) and true positives (tp)
    # by matching the predicted labels against the correct ones
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    
    specificity = tn / (tn + fp)
    sensitivity = tp / (tp + fn)
    
    print("{0:<35s} {1:6.3f}%".format('Specificity on ' + set + ' set:', specificity))
    print("{0:<35s} {1:6.3f}%".format('Sensitivity on ' + set + ' set:', sensitivity))

def plot_correlation(x):
    
    plots graph of correlation between features using matplotlib
    :param x: dataframe with features
    
    labels = x.columns.values
    fig = plt.figure(num='Features correlation')
    ax = fig.add_subplot(111)
    cax = ax.matshow(x.corr())
    fig.colorbar(cax)
    plt.xticks(range(len(labels)), labels, rotation='vertical')
    plt.yticks(range(len(labels)), labels)
    plt.show()

read_csv()
Baseline Neural Network

Below is the source code of the baseline Neural Network.

```python
import pandas as pd
from keras import Sequential
from keras.callbacks import EarlyStopping
from keras.layers import Dense, Dropout
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np

def read_data():
    # Reads data and stores it into dataframe df
    df = pd.read_csv('train_set', sep=',
    df_test = pd.read_csv('test_set', sep=',
    np.random.seed(21)
    split_dataframe(df, df_test)

def split_dataframe(df, df_test):
    # Splits dataframe into dataframe x, containing data for all features
    # and y, containing corresponding label (healthy or dementia)
    # :param df: whole dataframe for training set, before splitting
    # :param df_test: whole dataframe for test set
    
    # takes only dementia column (which are the labels, Y for dementia
    # and N for control)
    # and converts to numbers: 1 for Y and 0 for N
    y = df['dementia'].apply(lambda x : 1 if x == 'Y' else 0)
    y_test = df_test['dementia'].apply(lambda x : 1 if x == 'Y' else 0)
    
    # drops columns with empty values (pauses and retracing reform)
    # and drops dementia column because it is not needed
    x = df.drop(columns=['dementia', 'pauses', 'retracing reform'])
    x_test = df_test.drop(columns=['dementia', 'pauses',
    'retracing reform'])
    do_kfold_validation(x, y, x_test, y_test)
```
def do_kfold_validation(x, y, x_test, y_test):
    
    Perform kfold cross validation to avoid overfitting
    :param x: data from training set, before splitting
    :param y: labels from training set, before splitting
    :param x_test: data for test set
    :param y_test: labels for test set
    
    # initializes kfold with 5 folds, including shuffling,
    # using 9 as seed for the shuffling
    kfold = KFold(n_splits=5, random_state=9, shuffle=True)

    # splits datasets into folds and trains the model
    for train_index, val_index in kfold.split(x):
        x_train, x_val = x.loc[train_index], x.loc[val_index]
        y_train, y_val = y.loc[train_index], y.loc[val_index]
        normalize(x_train, y_train, x_val, y_val, x_test, y_test)

def normalize(x_train, y_train, x_val, y_val, x_test, y_test):
    
    Normalizes datasets to ensure that all features
    have mean = 0 and standard deviation = 1
    :param x_train: training set data
    :param y_train: training set labels
    :param x_val: validation set data
    :param y_val: validation set labels
    :param x_test: test set data
    :param y_test: test set labels
    
    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_val = scaler.fit_transform(x_val)
    x_test = scaler.fit_transform(x_test)
    train_model(x_train, y_train, x_val, y_val, x_test, y_test)

def train_model(x_train, y_train, x_val, y_val, x_test, y_test):
    
    Creates a 4-layer neural network and trains it
    :param x_train: training set data
    :param y_train: training set labels
    :param x_val: validation set data
    :param y_val: validation set labels
    :param x_test: test set data
    :param y_test: test set labels
    
    # create model
    model = Sequential()
    # input layer: 22 neurons just like the number of features
    model.add(Dense(22, input_dim=22, activation='relu'))
    # drops out to prevent overfitting
    model.add(Dropout(0.2))
    # first hidden layer
    model.add(Dense(12, activation='relu'))
    model.add(Dropout(0.2))
    # second hidden layer
    model.add(Dense(8, activation='relu'))
    model.add(Dropout(0.2))
    # output layer
model.add(Dense(1, activation='sigmoid'))

# compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# early stopping allows the model to stop training automatically
# when the chosen value (validation set accuracy) reaches its peak
# and starts decreasing
es = EarlyStopping(monitor='val_acc', patience=14)
history = model.fit(x_train, y_train, epochs=100, callbacks=[es],
validation_data=(x_val, y_val))
evaluate_model(model, x_train, y_train, 'train')
evaluate_model(model, x_val, y_val, 'validation')
y_pred = model.predict(x_train)
y_pred = transform_predictions(y_pred)
evaluate_sen_spec(y_train, y_pred, 'train')
y_pred = model.predict(x_val)
y_pred = transform_predictions(y_pred)
evaluate_sen_spec(y_val, y_pred, 'validation')
y_pred = model.predict(x_test)
y_pred = transform_predictions(y_pred)
evaluate_sen_spec(y_test, y_pred, 'test')
plot_history(history)

def transform_predictions(y_pred):
    """
    Scales each prediction > 0.5 to 1
and <= 0.5 to 0 to get binary values
:param y_pred: predicted probabilities
:return: transformed predicted probabilities
    """
    for i, v in enumerate(y_pred):
        if v > 0.5:
            y_pred[i] = 1
        else:
            y_pred[i] = 0
    return y_pred

def evaluate_sen_spec(y_true, y_pred, set):
    """
    Evaluates sensitivity and specificity
    on given dataset
:param y_true: correct labels
:param y_pred: predicted labels
:param set: type of set
    """
    # gets number of true negatives (tn), false positives (fp),
    # false negatives (fn) and true positives (tp)
    # by matching the predicted labels against the correct ones
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    specificity = tn / (tn + fp)
    sensitivity = tp / (tp + fn)
    print("\033[1;6.3f]\%".format('Specificity on ' + set + ' set:', specificity))
    print("\033[1;6.3f]\%".format('Sensitivity on ' + set + ' set:', sensitivity))
```python
def evaluate_model(model, x, y, set):
    
    Evaluates accuracy of the model
    :param model: trained model
    :param x: dataframe with data
    :param y: dataframe with labels
    :param set: type of set
    
    # evaluate the model
    scores = model.evaluate(x, y)
    print("{0:<35s} {1:6.3f}%".format('Accuracy on ' + set + ' set:',
        scores[1] * 100))
    print("{0:<35s} {1:6.3f}%".format('Loss on ' + set + ' set:',
        scores[0] * 100))

def plot_history(history):
    
    Plots history of accuracy and loss during epochs
    :param history: history of training (i.e. loss and
    accuracy throughout the epochs)
    
    # summarize history for accuracy
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'val'], loc='upper left')
    plt.show()
    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'val'], loc='upper left')
    plt.show()

read_data()

LSTM

The source code of the LSTM can be found in the dedicated GitHub repository:
https://github.com/eera-l/Geras
```