Information Extraction on Free-Text Sleep Narratives using Natural Language Processing

Fotedar, Sonali

Award date:
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Data Mining Research Group

Information Extraction on Free-Text Sleep Narratives using Natural Language Processing

Master Thesis

Sonali Fotedar

Supervisors:
Prof. Dr. Mykola Pechenizkiy (Eindhoven University of Technology)
Dr. Decebal Mocanu (University of Twente)
Mr. Dietwig Lowet (Philips Research)

Committee Members:
Prof. Dr. Mykola Pechenizkiy
Dr. Decebal Mocanu
Mr. Dietwig Lowet
Dr. Renata Medeiros de Carvalho

Final Version
Eindhoven, November 2020

Abstract

Recent years has seen a tremendous advancement in the field of Natural Language Processing. More and more research is focused on processing and understanding of unstructured or free-form text, primarily clinical and medical notes, due to their increasing availability and richness in information. However, not a lot of work exists on using NLP for understanding narratives related to personal sleep health. Sleep is one of the three pillars for a healthy life, and studies have shown that regular sleep tracking and identifying sleep issues can help in improving sleep health. In this work, we focus on developing techniques, for understanding narratives about sleep health of people, that can be adopted for digital assistance and sleep monitoring. We particularly focus on two types of narratives: narrative of people's last night's sleep and their description of sleep related issues. To extract structured information from the sleep narratives, we propose a novel pipeline that extracts temporal sleep events and outputs a complete structured timeline that can enable daily visual sleep tracking. We build our approach with three components. For the first component, Temporal Expression Extraction, we train a sequence tagging model that learns to tag temporal entities associated with sleep events using both word-level semantics and character-level morphological features. The second task of Temporal Expression Normalization is solved by training a specialised neural machine translation model that learns how to translate extracted temporal expressions into normalized time stamps. We adopt two strategies that we call Hybrid Tokenization and Context Conditioning that improve the translation results. Lastly, we adopt a rule-based Entity Parsing and Linking algorithm to process the extracted entities into a complete and structured timeline of sleep events. Moreover, we propose a novel architecture for multi-label text classification of the sleep issues. Our proposed method aims at capturing the semantic relation between the documents and the labels and leverages this relation to learn a document-aware label-to-label correlation. We achieve this by proposing a novel Graph Interaction-Attention Network for self-attention calculation. The results of our experiments show that our proposed method outperforms current state-of-the-art methods. Altogether, we present techniques to extract information from two types of sleep narratives that can be leveraged for building conversational assistants for monitoring sleep health and also contribute to the current research in solving the multi-label text classification problem.

Preface

This thesis marks the end of my two-year journey at the Eindhoven University of Technology. It would be an understatement to say that this journey, especially the last few months, has been one of the most challenging times of my life yet. However, each challenge gave me the opportunity to grown, personally and professionally, taught me great lessons and introduced me to some incredible people who joined me in facing these challenges together. Nevertheless, it has been a memorable and enriching experience overall, and I will always cherish it. I would like to use this opportunity to thank a few people who contributed to my journey in one one way or another.

I would like to start by thanking my supervisors; Decebal C. Mocanu at the University of Twente, for his invaluable guidance and support, and his unwavering faith in me even at times when I doubted myself, and Mykola Pechenizkiy at the Eindhoven University of Technology, for his esteemed supervision and keen insights. I would like to further thank Dietwig Lowet, at Philips Research, for his excellent mentorship during my internship. I would also like to thank Renata Medeiros de Carvalho for her valuable participation in my master's thesis defence committee.

I would like to thank my dearest friends, Selima, Berk, Gokhan, Irmak and Marta, who have been my family away from home. Thank you for giving me countless moments of joy and for being there with me through all the tears and laughs. I am so grateful for your love and support. Next, I want to thank my friends, Snehal, Shraddha and Shashank, who have been with me, in person or spirit, since the time I started my career in Computer Science six years ago. Even though we stay in different corners of the world, you three have managed to support me through my toughest times.

Lastly, but most importantly, I want to thank my loving family. To my partner, Surya, thank you for your endless love and patience. Even being apart all this time, you have continued to be my strength. To my little brother, Ayush, I live to be your biggest inspiration, and I hope I have made you proud. And finally, to my mom and dad, thank you for your unconditional love and support and immense trust in me. I am beyond grateful for all your hard work that has gotten me here. I cannot wait to see you guys again. I love you all.

Sonali Fotedar
Eindhoven, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of Natural Language Processing (NLP) in the past decades is backed by a couple of global developments, the universal hype around Artificial Intelligence, exponential advances in the field of Deep Learning and an ever-increasing quantity of available text data. However, most of these texts available are unstructured and in free-from. These unstructured texts contain an unreasonable amount of information that can be used to gain domain knowledge and bring quality solutions. One of the most prominent domains where information is largely found in free-text form, either typed or dictated, is the medical and health care domain. Health care research and operations consider free text medical and clinical notes to be a rich source of information. This has lead to extensive research into solving the challenging problem of automatically and accurately extracting the rich and nuanced information contained in free text.

The advancement in the field of Deep Learning in the recent years has lead to an upsurge in research in the field of Computational Linguistics. NLP techniques allow automatic processing of written human language through various tasks such as, text mining. These tasks are difficult for a computer to perform, but a particularly challenging problem is Natural Language Understanding (NLU). NLU, an integral part of any NLP task, is the ability of a computer to understand natural human language by simulating a human's ability to create natural language text. NLU uses algorithms to reduce human speech into a structured ontology. It is best to view NLU as the first step towards achieving NLP; the machine attempts to understand the language first, before processing it.

But why is it a particularly challenging problem for machines to understand natural language? We explain this by drawing upon the linguistic theory and decomposing the process of linguistic analysis into three dimensions: syntactic, semantic and pragmatic [73]. Human language is pervasively ambiguous and highly subjective owning to different perceptions of humans. It leaves room for common errors such as mispronunciations or transposed letters or words. Accessing language by aligning it with grammatical rules is a common way to derive meaning out of it, but grammatical correctness alone hardly ever validates human language. There is a need for a closer focus on multilevel semantics along with syntactic analysis to help the machine in understanding the meaning and interpretation of words. However, to fully understand the natural human language, the machine needs more than the literal meaning semantics provide. A pragmatic level of understanding to uncover the intent and context of

the text is necessary for completeness.

1.1 Business & Research Context

The thesis work presented here has been done in collaboration with Philips Research. Philips is a leading health technology provider focused on improving people's health and achieving better outcomes across the health continuum, from healthy living and prevention to diagnostics, treatment and home care. One of the leading research topics within Philips is Personal Health. In Personal Health, the focus is on investigating technologies and solutions that stimulate a healthy life, offering optimized personal care, and exploring ways to enhance a fulfilling home and interactive lifestyle. Among various domains that are researched, sleep health is one of them.

Sleep is vital for our well-being and is considered one of the three pillars for a healthy life. Even a few nights of poor sleep can have severe effects on aspects of daily life like alertness, memory, mood and cognitive function. While chronic sleep problems are often related to other health conditions like high blood pressure, depression or obesity [19], many sleep problems are caused by lifestyle and environmental factors. Some examples of these factors are disturbed sleep due to noise or light in the bedroom, caffeine consumption throughout the day, exercise, stress and irregular sleep cycle. Tracking sleep and identifying sleep issues and their causes can help to raise awareness of such problems and to take steps to improve sleep. However, people face certain barriers in their journey of improving sleep health that sleep tracking alone does not solve [53]. They fail to identify causes for their sleep problem if potential sleep contributing factors are not tracked. If people even identify the issues, they are unclear about what action they could take to combat these issues.

Therefore, one of the main focus of research at Philips towards improving sleep health is Digital Engagement. To enhance consumer engagement, Philips aims at offering digital interactions as fully personalized experiences to each consumer. This serves as the basis for the motivation to work towards a voice-driven conversational agent for sleep tracking and dialogue-based sleep problem assessment. Current solutions within Philips only make use of structured data. The broader goal is to have a conversational agent that can process and understand unstructured form of data, especially in the form of speech for ease-of-use of the consumer, and extract useful information that can help towards bringing solutions to the user and improve their sleep health. Therefore, in this thesis, we focus on NLP based methods of information extraction from sleep narratives.

1.2 Problem Formulation

Our aim with this thesis is to develop information extraction techniques that can process free-text sleep narratives and help in monitoring sleep health. We deal with two types of sleep narratives: accounts of the users' last night's sleep, where they mention the events of their night specifying the time they went to bed or sleep, if they were disturbed, when they eventually woke up and details of other sleep events during the night; and narratives of people describing their sleep issues such as problems with staying awake, falling asleep or snoring. Our problem at hand is two-fold and hence, we tackle it as two distinct tasks.

1.2.1 Sleep Timeline Extraction

Our first task is to extract a complete sleep timeline from the sleep narratives provided by the user. This narrative would essentially be retrieved from the dialogue between the user and the conversational agent and is either typed or dictated and is essentially in amorphous form. We tackle this problem by breaking it down into sub-processes and simultaneously give a formal definition of the problem:

1. Detecting temporal expressions and their related sleep event: Given the user's textual narrative of their sleep last night, the goal is to detect temporal expressions (or entities) in the text and tag them with the type of sleep event they are associated with. This problem can be modelled as a sequence tagging problem. Given a sequence of tokenized text that represent the sleep narratives, we want to learn a function that can tag token or spans of tokens that represent time expressions and tag them with the associated sleep event. Eventually, we want to train a Named Entity Recognition (NER) model that can learn to identify these temporal entities in the text and classify them into pre-defined sleep events classes.

2. Normalising the temporal expression: As we mentioned before, human language is ambiguous and hence leads to various representations of similar concepts. Same is the case with temporal expressions. Different people write or mention time differently based on their preferences. We need to formally define different kinds of temporal expressions and learn a function that can translate different types of temporal expressions and different representations of the same time into one normalized structure so they can be easily manipulated further. This can be framed as a Neural Machine Translation (NMT) problem where the model takes the chunks of temporal entities previously extracted from the text and learns to translate it into a normalized form.

3. Parsing the extracted entities into a complete timeline: Users' account of their sleep is often not complete since they do not always recount every detail. However, it is important to have a complete timeline, specially if we want to visualize it for sleep tracking. Hence the extracted temporal entities need to be parsed into a structured and complete timeline.

1.2.2 Identifying Sleep Issues

In addition to sleep timeline extraction, we focus on our second task of detecting sleep issues and causes from the user's dialogue. Based on the sleep issues text corpus, we pre-define categories of sleep issues and formulate this as a Multi-Label Text classification (MLTC) problem. Since label classes co-exist in MLTC problems, there is an inherent correlation between them that is often not very apparent. Current research in MLTC works towards learning these correlations in order to build better solutions for multi-label classification. We continue on this path of the current research and focus on solutions that take into account the correlation between labels.

When compared to binary or multi-class classification, multi-label classification faces an issue of estimation of posterior probabilities during evaluation. The most common method is to use a threshold. Several threshold selection strategies exist however they require cross-validation

and human input. This is not scalable when we have a large MLTC problem with a growing number of labels. Exploratory data analysis on the corpus (please refer to Section 4.1 for more details) also reveals that a large percent of the data samples are classified to only one label while the rest are classified to multiple labels. Training on this dataset will lead to the model assigning a very high probability score to only label, and low probability scores to the other labels. Finding an optimal threshold, in this case, becomes more challenging. Hence, we focus on finding a suitable alternative that can overcome these problems.

Classifying issues will enable us to build a recommendation system that can recommend articles and tips in how to combat the issues they are facing thereby breaking the barrier we discusses in Section 1.1. However, we do not focus on building a recommendation system in this thesis due to current lack of corpus of tips and helpful articles to combat sleep issues and leave this as future work.

1.3 Research Questions

We define three research questions addressed in this thesis:

1. Can we develop a solution that can understand an individual's sleep narrative and extract a complete and structured timeline from it? Can this solution deal with the inherent ambiguity in the human language?

2. Can we propose a solution for our sleep issues MLTC task that advances the current state-of-the-art by taking into account the correlation of labels and words?

3. Can we conceive an alternative solution for estimation of posterior probabilities from the MLTC model that requires no overhead of cross-validation and is scalable? Will this solution handle the skewed distribution of the dataset?

1.4 Contributions

Overall, the objective of this thesis is to extract information from two types of sleep narratives for the purpose of building a voice-driven agent that helps with sleep monitoring and sleep health. The main contributions of our work are as follows:

1. We present a novel pipeline, Sleep Events Recognition and Normalisation (SERN), that detects sleep related temporal entities in free text and extracts a complete and structured timeline from the temporal entities.

   (a) We develop a design for the Temporal Expression Extraction task that leverages two state-of-the-art word-level and character-level embeddings for training a NER model for extraction of temporal events.

   (b) We propose two strategies, namely *Hybrid Tokenization* and *Context Conditioning* to improve the performance of the Transformer model that we adopt for the Temporal Expression Normalization task.

2. We propose a novel method called Multi-label Text Classification using Graph Interactive-Attention Network, for performing MLTC tasks. Our method is based on a novel architecture we introduce, called the Graph Interaction-Attention Network that takes into account the correlation between words and labels to further learns document-aware label-label correlations for better prediction.

3. We adopt a new method for estimation of posterior probabilities that uses the pooled features from BERT to predict the number of labels each instance of the data should be classified into. We train a linear layer with a multi-class classification objective to achieve this.

1.5 Thesis Outline

This chapter gave an introduction to Natural Language Processing and the Business and Research Context of our research. We further formulated our problem, posed our research questions and listed our main contributions. Chapter 2 gives some background on NLP over the years and presents some related work done in the field. Chapter 3 describes our proposed pipeline SERN for extracting structured timeline from free-text sleep narratives and our novel methodology, MLTC-GIN to tackle the MLTC problem using Graph Interactive-Attention Network to learn label correlations that are also document aware. Chapter 4 gives details about the datasets used in our study along with the experimental setup used for evaluation. We further present the results of our experiments in Section 4.3 that include performance comparison and ablation studies and simultaneously discuss the outcomes. Lastly, Chapter 5 presents our conclusions, limitations of our work and suggestions for future works that can follow.

Chapter 2

Background & Related Work

In this chapter, we briefly focus on the core trends in NLP over the years, how the advances in the field of Deep Learning affected NLP and its recent inflection points. We further discuss some related works on natural language processing of clinical narratives, temporal information extraction and normalization and multi-label text classification.

2.1 NLP Over The Years

In the core of many natural language processing tasks is Language Modelling. Given a vocabulary of words, a Language Model learns the likelihood of occurrence of a word based on the previous sequence of words used in the text. Mathematically, they provide the conditional probability of the next word, given some previous words in the sequence. Classic approaches to language modelling are based on n-grams and employ smoothing to deal with unseen n-grams [48]. The first neural language model based on a feed-forward neural network was proposed in 2001 [6].

2.1.1 Word Embeddings

For a long time, most techniques used to solve NLP tasks made use of hand-crafted and time-consuming features. These high-dimensional, sparse vector representations of text, for example bag-of-words model, lead to the problem of *Curse of Dimensionality*. Dense and low dimensional representations of words or word embedding were introduced as early as 2001, but the real breakthrough was witnessed with the introduction of Word2vec [62]. Offered in two flavours, Skip-gram and Continuous Bag-of-words models, they gained popularity as they could learn relatively high-quality word embedding with semantic information by performing large scale training. Improvements over word2vec have also been introduced, such as skip-gram with negative sampling and subsampling of frequent words [64]. While it captures intuitive relations between words, word2vec also suffers from several limitations. It cannot handle unknown and unseen words and cannot present phrases. It also does not take into account polysemy and is not robust towards biases that may come through training data. It is only able to learn word representations through the local context and lacks a global context.

Over time, much work has been done in studying different aspects of word embeddings and how to overcome past limitations. Character level model was introduced to learn morpholo-

gical information in words such as the characters and their combinations [47]. GloVe word embeddings are another popular alternative to word2vec that uses matrix factorization to learn representations of the words [77]. Unlike word2vec, it does not only rely on local context window for local statistics but also uses global word co-occurrence to obtain word vectors.

2.1.2 First Inflection Point: Deep Learning

Soon after the reintroduction of neural networks and the rise of deep learning, neural network models became a prime choice of method for NLP. Recurrent Neural Networks were the first obvious choice due to their dynamic temporal behaviour [63]. They were quickly replaced by models that were more resilient towards the problem of vanishing and exploding gradients such as, Long-Short Term Memory (LSTM) models [35] and Gated recurrent Units (GRU) network. Bidirectional LSTMs or BiLSTMs are also used when context on both the left and the right side of the word is essential and effectively increases the amount of information available to the network [34]. Convolutional Neural Networks (CNNs) have been widely successful for computer vision due to their ability to recognize shapes and patterns, and they have also proved to be effective at NLP tasks as well [43, 46]. Their particular advantage over LSTMs is that they can be parallelized and studies have used convolutions to speed up LSTM models [10].

Sequence to sequence or Seq2Seq learning was introduced in 2014 for mapping one sequence to another [94]. The framework consists of two neural models, encoder and decoder, where the encoder processes the sequence token by token and outputs a compressed and information-rich vector representation and the decoder takes this as the initial state and predicts the output, token by token while taking the last predicted token as input. This framework was found to be a perfect fit for Machine Translation. Different models can play the role of encoder and decoder. The decoder can be conditioned on arbitrary representations instead of the sequence, and due to this flexibility, it has found itself many applications such as image caption generation [102] and language translation [25]. [18] proposed RNN encoder-decoder model that introduced the Gated Recurrent Units (GRU) instead of LSTMs for statistical machine learning.

2.1.3 Second Inflection Point: Attention Mechanism and Pre-training Language Models

The core of Neural Machine Translation (NMT) and the foundation for the next inflection point in NLP are Attention networks introduced by [5]. The major drawback of previous seq2seq models is that the entire context of the input sequence was condensed in a vector. Attention mechanism overcomes this by allowing the decoder to look back at the hidden states of the input sequence and leverage their weighted average as additional input. [5] proposed the RNNsearch model that used their proposed attention mechanism to reduce "information compression" in a single vector and instead allow the decoder to look at the entire source sentence (via its hidden states) at each decoding step. Attention is available in different forms [60] and has a wide range of applications such as reading comprehension, one-shot learning and image captioning, to name a few. The use of attention mechanism in NLP systems offers

a great side effect by providing a glimpse into the inner workings of the model. It highlights different parts of the high-dimensional input that are relevant for a particular output based on the attention weights providing a model-agnostic local explanation [83]. Techniques such as LIME [87] and SHAP [58] have been recently used to interpret NLP models.

Transformer is a simple network architecture based on the attention mechanisms that tackled the problem of machine translation [100]. The model is made up of an encoder and decoder component where each is a stack of the same number of encoder and decoder blocks, respectively. The encoder's inputs first flow through a self-attention layer, and then the outputs are fed to a feed-forward neural network. The same feed-forward network is independently applied to each position. The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence (similar to what attention does in seq2seq models).[1] The paper also introduces a very important refinement on the self-attention mechanism, called Multi-head Attention. Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel. The paper states that "*multi-head attention allows the model to jointly attend to information from different representation sub-spaces at different positions. With a single attention head, averaging inhibits this.*"

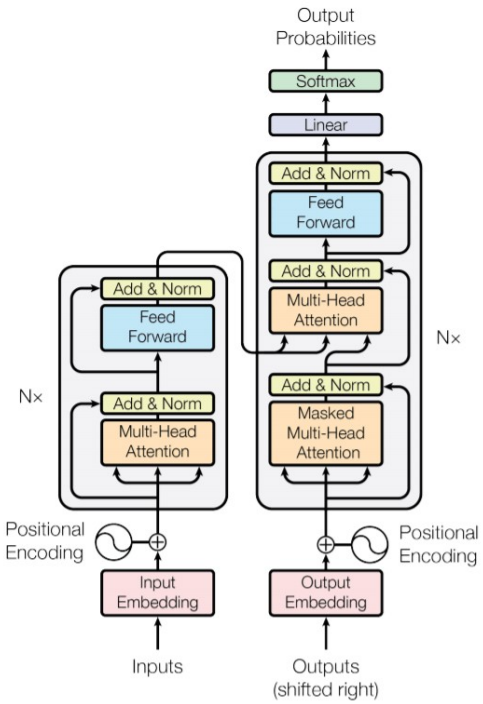

Figure 2.1: The Transformer model architecture [100]

[1] An illustrated explanation of the Transformer model can be found here http://jalammar.github.io/illustrated-transformer/

The shift from pre-training word embeddings to pre-training language models was the final catalyst that lead to the inflection point in NLP. Since language modelling is an unsupervised task, their training can thus be scaled to billions of tokens, new domains, and new languages. It was first proposed in 2015 [23] and soon proved to be beneficial for a wide range of tasks. Pretrained language models have been used to derive deep contextualized word representations for target model [78] or were fine-tuned on supervised learning tasks [84], with study showing these contextualized embeddings giving large improvements over the state-of-the-art across several different tasks [37].

The OpenAI Transformer or GPT [2] combined two powerful ideas: the Transformer model and unsupervised pre-training, resulting in a breakthrough scalable, a task-agnostic system that achieved state-of-the-art performance on various language tasks [81]. Their goal of learning a universal representation by large scale unsupervised learning that transfers with little adaptation to a wide range of tasks by fine-tuning was adapted by other studies resulting in many powerful language models like the GPT2 [82], BERT [24] and GPT3 [11].

BERT has attained state-of-the-art performance in several NLP tasks. The core of BERT is the Transformer encoder, but its critical technical innovation is the bidirectional training of the language model. It proposed the novel technique of Masked Language Modelling (MLM) that attains a profound sense of language context and flow as compared to unidirectional language models [3]. Development of the BERT model has also lead to several spin-off models with improvements such as DistilBERT [89], a distilled version of BERT, a robustly optimized BERT called RoBerta [56] and ALBERT [50], lite BERT for self-supervised learning. [2] introduces the novel *contextual string embeddings* or FLAIR embeddings that leverages the internal states of a trained bidirectional character language model and significantly outperform previous works on English and German named entity recognition (NER).

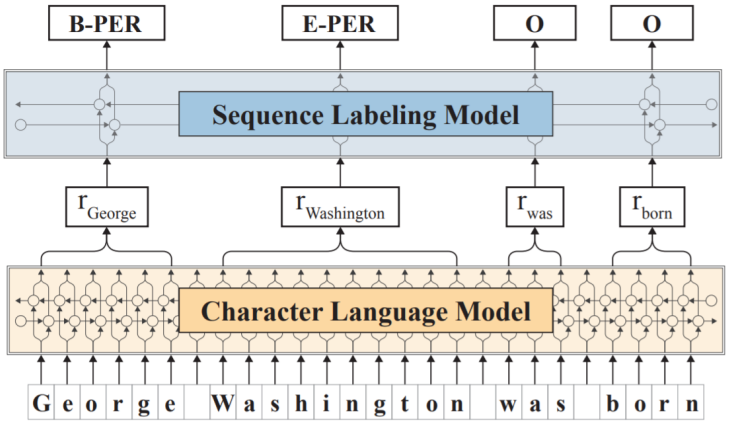

Figure 2.2: Named Entity Recognition using contextual string or FLAIR embeddings [2].

[2] Generative Pre-Training
[3] For a detailed explanation of BERT refer to http://jalammar.github.io/illustrated-bert/

2.2 NLP on Clinical Narratives

Clinical narratives are the most dominant form of communication within the health care discipline; it allows for personalized account of the patient's history and contains rich information that is essential for clinical assessment and decision. Traditional NLP methods, especially rule-based approaches, have long been used to extract knowledge from these texts [91]. Use of these methods generally requires interaction with experts to gain domain knowledge. However, cutting-edge methods employing computational linguistics and machine learning have proved themselves to be efficient and fruitful in dealing with clinical texts. Although machine learning-enabled NLP solutions require a significant amount of annotated data to learn from, it is argued that the time required to annotate this data for training may require the same time to extract knowledge [36, 72].

A large number of tools and frameworks exist for general-purpose information extraction from clinical dictionaries, such as cTAKES [90], NOBLE [98] and MedLee [28]. Most of the earlier NLP methods used on clinical narratives are either rule-based approaches, traditional machine learning-based approaches or a hybrid of them. Rule-based approaches include methods that include a dictionary look-up [114, 70], identifying terminologies based on domain ontologies [66, 1], different manually written rules [14, 97] and regular expression patterns [68, 105]. One of the most widely used machine learning techniques is Support Vector Machines (SVM) used for mortality prediction [30], detecting axial sPA from EHRs [104]. Naive Bayes is another popular traditional method employed for heart disease prediction, classifying smoking status and EHRs for obesity [26]. CRFs based approaches have been used frequently on clinical narratives for Named Entity Recognition on disorders [51], identifying EHR notes concerning diabetes [13]. Lastly, Random Forest was used for classification of notes for pain assessment [27], cancer type classification [44] and predicting heart diseases [106].

Recently, studies have been employing deep learning-based methods for various tasks on clinical narratives. Attentional convolutional neural networks have been used to extract International Classification of Diseases (ICD) codes from clinical notes [112]. Writing styles of clinical notes can vary greatly, and transformer-based architecture with generative modelling algorithms were applied to tackle this issue [67]. Deep neural networks have also been applied to unstructured text notes for phenotyping youth depression. Several studies focus on processing Electronic Health Records (EHRs) by using language model pre-training to improve on hierarchical patient classification [45], learning the implicit graph structure of EHRs [20, 21] or extracting features from the records and automatically predicting future risk [69]. [52] pre-trained a domain-specific language representation model, BioBERT, on large-scale biomedical corpora which largely outperforms BERT and previous state-of-the-art models in a variety of biomedical text mining tasks.

Rule-based regular expression method have been used to extract sleep data from free-text notes [71] and identify sleep disorders [16]. Despite the growth of deep learning based solutions for the processing of various types of unstructured clinical narratives, not a lot of work has been done in processing on narratives related to sleep. In this work, we propose a novel pipeline that combines pre-trained language modelling, deep learning and rule-based method to extract temporal information from free-text sleep narratives and construct a complete timeline from the temporal information.

2.3 Temporal Information Extraction & Normalization

One of the critical aspects of clinical or study notes that is of interest is timeline. Extracting a patient's timeline can prove to be very beneficial in tracking progress, longitudinal effects of treatment and predicting clinical outcomes. Temporal Information Extraction has been long studied but was mainly addressed in the news domain. In the early 2000s, TimeML marked a major milestone in the discipline as the first full temporal annotation scheme [80]. TimeML is a robust specification language for events and time expression with the primary purpose of identifying and extracting events and their temporal anchoring from a text. The interest in temporal extraction in medical domain developed with the introduction of the i2b2 NLP Challenge in 2012, providing a corpus of discharge summaries annotated with temporal information [93]. Several approaches have been devised in the literature for temporal information extraction ranging from rule-based methods to fully supervised machine learning approaches to deep learning approaches.

Temporal Extraction

Rule-based approaches have proved to be efficient in extracting temporal entities due to the low diversity of realizations in the text. HeidelTime, a multilingual, domain-sensitive temporal tagger, is one of the most popular rule-based approach [92]. It extracts temporal expressions from documents and normalizes them according to the TIMEX3 annotation standard. Another rule-based approach, SUTime, was introduced by the Stanford NLP Group for recognizing and normalizing time expressions and is part of the StanfordCoreNLP pipeline [12]. However, these approaches suffer from a limitation that they fail to detect temporal expressions when they have not been explicitly expressed, or they are expressed in relation to another temporal entity. They also fail to detect explicitly written temporal expression when there is absence on indicators usch as *"AM"*, *"PM"* or *"o'clock"*.

Machine Learning approaches tackle this task by solving it as a sequence labelling task. Conditional Random Fields (CRF) is a very popular probabilistic model often applied in pattern recognition and sequence labelling. [42] tackle the TempEval time expression extraction task by modelling it as a sequence labelling problem and using CRF to label each token in the corpus. [7] uses a similar approach on the same shared task but instead employs the SVM model. Bidirectional LSTM-CRF (BiLSTM-CRF) model was introduced by [40] for sequence tagging that combines the complexity, non-linearity and bidirectional property of BiLSTMs with the ability of CRFs to capture the relationships at label level and predicting optimal, joint prediction of all labels. BiLSTM-CRFs have since been widely used by studies for sequence tagging tasks, specifically Named Entity Recognition (NER) [59, 79, 54]. NER is the subtask of Information Extraction with the goal of locating and identifying names entities in free text into pre-determined classes. It has also been used to identify time and date entities. SpaCy, an open-source library for advanced Natural Language Processing in Python, provides support for NER. SpaCy's NER tool is trained on the OntoNotes v5.0 corpus [107] and can detect 18 different entities including time (times shorter than a day) and date (absolute or relative dates or periods). As mentioned in Section 2.1.3, FLAIR embedding has shown impressive results on NER tasks, and experiments showed significant performance improvement when BiLSTM-CRFs were used as the sequence tagging model as compared to a feed-forward linear architecture or a feed-forward linear architecture with CRF [2].

Temporal Information Normalization

Rule-based approached, HiedelTime and SUTime map detected time entities into TIMEX format. However, identifying temporal entities using sequence labelling requires an additional step of normalizing the time. One of the first approaches towards normalizing time expressions was TempEx [61] and was later extended to handle time expressions based on the TimeML TIMEX3 standard and released as GUTime. TIMEN was introduced as a framework for time expression normalization making use of Knowledge Base (KB) and rule matching [57]. Machine learning-based related works on time expression extraction discussed in the last section use TIMEN for time normalization.

Events corresponding to the time entities are generally extracted using event extraction methods [38]. However, in our work, we jointly extract the time and corresponding event by identifying time entities and classifying them into pre-determined event classes (refer 3.1.1). Our approach is suitable for our study due to our niche domain and a definite number of events. Moreover, time expression normalization methods discussed above fail to normalize time when time indicators AM or PM are not mentioned, or they fail to detect expressions that express relative time duration, for example, *"after 10 minutes"*. They also fail to handle misspelt words and abbreviations. We address these issues by taking a deep-learning approach by training a model that can learn these rules but can also generalize on new unseen examples and be robust to misspellings and errors. Hence, we employ an attention-enabled sequence-to-sequence model trained on our artificial dataset to translate time expressions into normalized versions (refer 3.1.2).

2.4 Multi Label Text Classification

Multi-Label Text Classification (MLTC) is the task of classifying each data sample in the corpus into one or more categories. This makes it both a challenging and essential NLP task. MLTC has been used in a wide variety of settings such as, sentiment analysis [55], tag recommendation [29] and information retrieval [33] to name a few. MLTC tasks have been using traditional machine learning methods as well as deep learning-based methods. We present these related works by categorizing them into two following types.

2.4.1 Problem Transformation Methods

Problem transformation methods handle the MLTC problem by dividing the multi-label problem into one or more conventional single-label problems and then applying any single label classification algorithm. The baseline approach is Binary Relevance (BR) [9], which transforms the multi-label classification problem into several single-label classification problems. This approach is problematic since it completely ignores the correlation between the labels. Another problem transformation approach is Label Powerset (LP) [99]. LP transforms the multi-label classification problem to a multi-class problem by creating a binary classifier for every label combination present in the training set. LP considers label correlation but still suffers from several limitations such as the problem of label imbalance, especially when the number of distinct label sets is high compared to the number of instances in the dataset. LP can only predict label combinations that occur in the training set and therefore would have to be trained with an addition binary classifier each time a new combination is observed. The

complexity of both BR and LP increases in the case of a huge number of labels.

Another transformation method is the Classifier Chains (CC) [86], which transforms the multi-label problem to a chain of single-label problems. Each model predicts in the order specified by the chain using all of the available features provided to the model plus the predictions of models that are earlier in the chain. This approach is similar to BR but takes label correlation into consideration. CC still suffers from a drawback since it depends heavily on the order of the chain. Different order of the chain results in a difference in prediction and the performance of the model. This limitation was tackled by Ensemble of classifier chains [85] that randomly orders the classifier chains.

2.4.2 Problem Adaptation Methods

Algorithm adaptation adapts the algorithms to handle multi-label data directly, instead of transforming the data. The popular C4.5 algorithm [88] was adopted by [22] to handle multi-label classification by constructing a decision tree and developing re-sampling strategies. SVM ranking was adapted to create a multi-label ranking algorithm called Rank-SVM [41] although it does not account for label correlation. [111] adapted the popular K nearest neighbour algorithm [4] and proposed a lazy learning approach called ML-KNN. However, the algorithm is limited to utilizing only the first or second-order correlation between labels.

2.4.3 Neural Network Models

More recently, different neural networks have been adopted to solve the MLTC problem. Back Propagation for Multi-Label Learning (BP-MLL) [65] adapted traditional feed-forward network and optimized a loss function similar to ranking loss. [110] proposed a model, Hierarchical Attention Network (HAN) that uses GRU gating mechanism with hierarchical attention. However, these models perform poorly on high dimensional, large-scale data.

Several studies used the CNN [46] and LSTM architecture for multi-label classification tasks and achieved state-of-the-art performance; however, they use logistic regression for each label independently and fail to learn label difference. Recurrent Convolutional Neural Network (RCNN) [49] and sequence learning models like Ensemble model CNN-RNN [17] and Sequence Generation Model (SGM) [108] that generates possible label classes using an RNN-based decoder have been adopted for MLTC tasks. However, they neglect the correlation between labels.

Several works have been focused on taking label correlation into account. Works by [32] and [31] leverage the hierarchical and graphical structures among labels. [113] proposed hierarchical SVM with orthogonal transfer to learn the hierarchy in labels, [76] use graph networks to get the graph-of-words representation of text to tackle the hierarchical MLTC problem. These methods take into account the label correlation at only at pair level.

BERT language model [24] achieves state-of-the-art performance in many NLP tasks and can be adapted to perform MLTC. [74] proposed an attention-based graph neural network, MAGnet. They use Graph Attention Network (GAT) [101] to capture the attentive dependency structure among the labels. [39] proposed a hybrid attention neural network, LAHA,

that considers both document content and label structure to create a label aware document representation. They introduce an interaction-attention mechanism to explicitly compute the semantic relation between the words and labels. We take inspiration from the MAGnet and LAHA models and propose a novel framework, MLTC-GIN, to solve our problem of MLTC on sleep issues corpus.

Compared to binary or multi-class classification, multi-label classification faces an issue of estimation of posterior probabilities during evaluation. The most common method is to use a threshold. Studies have proposed several threshold selection strategies such as PCut, SCut and RCut [3, 109]. However, these parametric strategies have an overhead of expensive cross-validation or human involvement. We adopt a simple alternative for the estimation of posterior probabilities that requires no overhead of selecting this parameter via validation. It turns out our idea closely resembles the work done by [96] where they use an OnevsRest SVM to get the score denoting the class membership of each data point and use another OnevsRest SVM to predict the number of labels based on the scores. However, there have been incredible advancements in NLP techniques since then, such as the BERT language model showing state-of-the-art performance. Therefore, we propose to use the pooled features extracted from BERT to train a linear layer to predict the number of labels for each instance.

Chapter 3

Proposed Methodology

In this chapter, we present our proposed methodologies to advance the natural language understanding of sleep narratives.

1. We address the timeline extraction task by proposing a novel pipeline, Sleep Events Recognition and Normalization. The pipeline takes a divide-and-conquer approach by splitting up the problem into two key sub-problems: temporal expression extraction and temporal expression normalization. For each sub-problem, we propose a specialized machine learning approach. We also write a rule-based algorithm to parse the extracted temporal entities into a complete structured timeline.

2. We also propose a novel graph interaction-attention network based multi-label text classification model that accounts for the semantic connection between words and labels. We also adopt a simple strategy for evaluation of posterior probabilities that overcomes the drawback of using a threshold.

3.1 Sleep Events Recognition and Normalization

Given the user's description of last night's sleep, our proposed pipeline uses a combination of deep learning-based and rule-based methods to extract a complete timeline of the sleep events. First, the free-text narrative of the user's last night's sleep is fed into a time expression extraction model that extracts nine classes of sleep-related time events that may be present in the text. This is tackled as a named entity recognition task. Next, the extracted time entities, along with their class category, are fed to a time normalization model that translates the time into a normalized form in 24-hour format. The extracted and normalized time expressions from the given text are then parsed, making sure there are no gaps in the timeline and non-explicit type of time expressions are converted to explicit type by using a rule-based algorithm. Finally, the entities are linked, forming a complete timeline. Figure 3.1 shows a high-level overview of our approach.

Before we dive deep into the details of our approach, we define the set of temporal events that we want to focus on and subsequently use as categories for temporal tagging.

- BED_TIME: This is the time the user goes to bed or their bedroom for the intention of going to sleep.

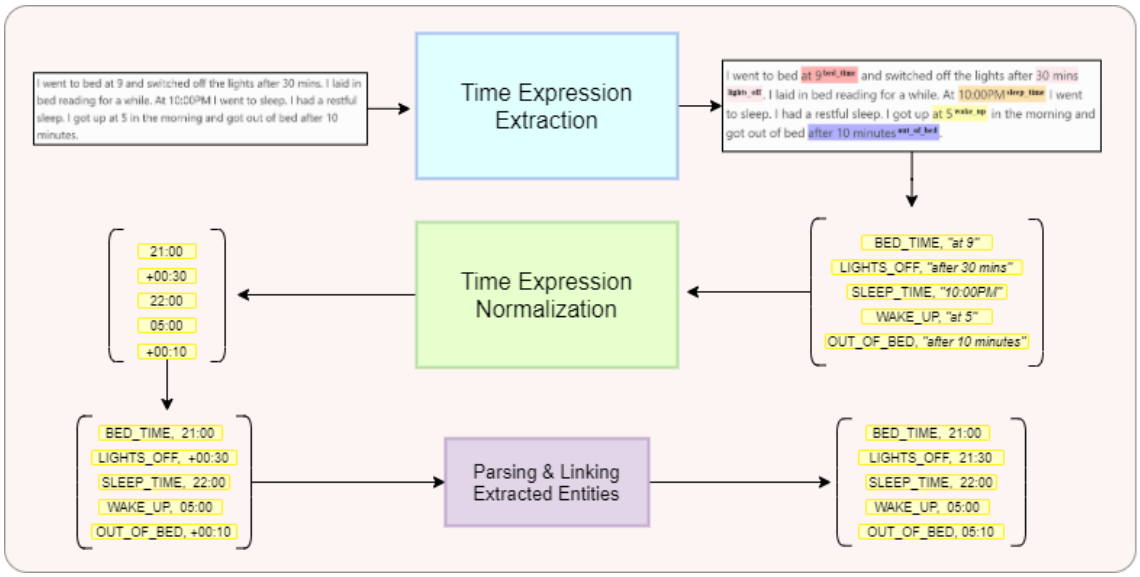

Figure 3.1: High-level Architecture of the Sleep Events Extraction and Normalization pipeline.

- LIGHTS_OFF: This is the time the user switches off their light before going to sleep.
- SLEEP_TIME: This is the time when the user actually falls asleep.
- SLEEP_LATENCY: This is the duration of time it takes for the user to go from being fully awake to sleeping.
- SLEEP_DISTURBED: These are the times when the user's sleep is disturbed either naturally or due to external factors like noise, light or other individuals.
- DURATION_OF_DISTURBANCE: This is the amount of time the user was disturbed from their sleep.
- WAKE_UP: This is the time the user finally wakes up from their sleep intending to get up from the bed and start their day.
- OUT_OF_BED: This is the time when the user finally gets out of the bed.
- SLEEP_DURATION[1]: This is the duration of time the user was asleep.

We discuss each component of the approach in the following sections. It is important to note that for the scope of this thesis, we only focus on sleep descriptions of the individual's previous night's sleep and therefore, we focus on temporal entities at day-level and not focus on higher level temporal entities such as day, month or year.

[1]This time event is extracted for keeping statistics and does not play a role in creating a timeline.

3.1.1 Temporal Expression Extraction

We use a combination of contextual word-level and character-level embeddings, namely BERT [24] and FLAIR embeddings [2], that is pre-trained on a large corpus. Each sentence in the training set is passed to the bidirectional LSTMs-based character-level neural language model. For every word, the internal character states are retrieved to form the contextual string embeddings. From various contextual embeddings discussed in Section 2.1.3, we adopt the FLAIR embeddings for our approach due to their use of character-level features. [2] state that pre-trained contextual character-level features are particularly helpful for NER tasks since entities are an open vocabulary of names that are often indicated by character features (such as capitalization or endings), as well as their contextual use. We further hypothesis that pre-trained contextual character-level features as opposed to using only contextual word-level features are particularly suitable since they can learn the useful orthographic and morphological structure of numeric characters in time expressions. For our pipeline, we stack the FLAIR embeddings with BERT embeddings to add greater latent word-level semantics. This eventually means that the embeddings obtained from both FLAIR and BERT are concatenated for each word and passed through a linear layer to obtain the final feature representation vector.

The final feature vector is passed to a BiLSTM-CRF for sequence labelling. The BiLSTM takes the embedddings, and the forward and backward output states of the BiLSTM are concatenated and fed to the CRF. We adopt BiLSTM as they have shown to be effective for various sequence labelling tasks. In contrast to a linear classifier that models the product of the conditional probability of a label given a token for each token in the sequence, CRF models the conditional probability of the entire sequence of labels y given the entire sequence of tokens X. This is because the regular classifier assumes that there is no dependence between adjacent positions in the sequence. Let r_i^f be the forward output state and r_i^b be the backward output state for the i_{th} word, where $i \in [1, 2, 3, ..., n]$ and n is the number of tokens. Then, the input to the CRF is given by:

$$r_i := \begin{bmatrix} r_i^f \\ r_i^b \end{bmatrix} \tag{3.1}$$

The final sequence probability is then given by a CRF over the possible sequence labels y (Equation 3.2). Finally, the Viterbi algorithm [103] is adopted for decoding the optimal output sequence.

$$\hat{P}(\mathbf{y}|\mathbf{r}) = \prod_{i=1}^{n} \exp(\mathrm{W}_{y_{i-1},y_i}\ r_i + \mathrm{b}_{y_{i-1},y_i}) \tag{3.2}$$

where, W_{y_{i-1},y_i} is the $m \times m$ transition matrix with m being the number of labels in the training set and gives the cost of transition from label y_{i-1} to label y_i. We choose to adopt a CRF later because the CRF layer considers the correlations between the current label and neighbouring labels and adds syntactic constraints to the final predicted labels to ensure they are valid. These constrains are learned by the CRF layer through the transition matrix automatically from the training dataset during the training process.

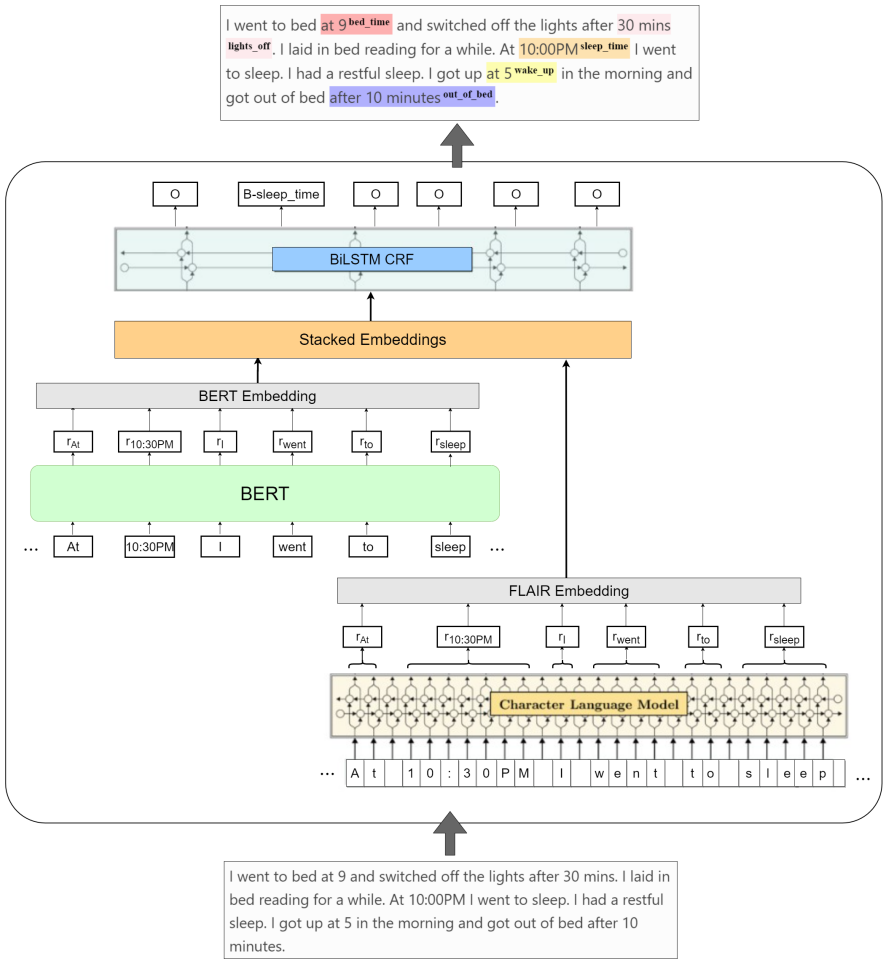

Figure 3.2: Architecture of the method adopted for the Temporal Expression Extraction task: two embeddings BERT [24] and FLAIR[2] are stacked to extract word-level and character-level features respectively. BiLSTM CRF model is used to tag the sequences.

3.1.2 Temporal Expression Normalization

The previous component of our pipeline detected the temporal entities in the text and predicted the sleep event it was associated to. Since the detected temporal entities are still spans of the original text, representation of the same time can vary across different data samples due to different ways humans use language to represent time textually. For example, 21:00 can be expressed in different ways such as 9 PM, 9 o'clock or 9.00 p.m., depending on the individual's preference. Moreover, established rule-based algorithms fail to detect, and subsequently, normalize time expressions when they are vaguely defined or when indicators AM/PM is not explicitly mentioned. Therefore, it is essential to translate these temporal entities into a normalized form.

Normalisation Taxonomy

Establishing a taxonomy of time expressions is an essential step before going into details of our methodology. We define five types of time expressions observed when users' describe their sleep:

- **Explicit:** Time expressions are stated explicitly in terms of hours and/or minutes and can be directly translated to `HH:MM` format. *For example, "10:30", "8 pm", "at 9".*
- **Durative and/or Relative:** Time expressions that are expressed in duration either in absolute form or relative to another time expression. These are normalized as `+HH:MM` or `-HH:MM` with the `+/-` indicating the relative nature meaning that this duration of time needs to be added or subtracted to a relative time to obtain the exact time of the event. *For example, "after 10 mins", "two hours", "40 minutes later".*
- **Frequency:** Time expressions that are durative and regularly recurring. These are normalized as `*HH:MM` with the `*` indicating the recurring nature of the expression. *For example, "every 5 minutes", "every half an hour".*
- **Range:** Time expressions that are expressed as range between two explicit time expressions. These are normalized as `HH:MM-HH:MM`. *For example, "from 2:00 to 4:00 am", "9 - 10 pm".*
- **Quantitative:** These expressions are particularly used to define the number of times an event occurred. These are normalized as `T`n, where n is the number of times the event occurred. *For example, "5 times", "2-3 times".*
- **Vague:** These are durative time expressions are used when the user is unsure about the time or duration. These are normalized as `+HH:MM` and we use heuristics to deal with them. *For example, "a few minutes later", "a couple of hours".*

We adopt the neural machine translation (NMT) approach for our temporal normalization task. We create an artificial dataset by generating different time expressions for each sleep event by defining some regular expressions based on common representations of time as used by humans. For each data sample, we also generate its normalized time form in 24-hour format as ground truth for our training. To define this more formally, let $\{(t_i, c_i, \hat{t_i})\}_{i=1}^{n}$ be our set of labelled training data, where t_i is the i^{th} time entity, c_i is the sleep event category associated with the i^{th} time entity. In the scope of this component of the pipeline, we also call this as the *context* to the NMT model. And finally, $\hat{t_i}$ is the i^{th} normalized time entity.

We use the attention-based sequence-to-sequence model, Transformer by [100]. The input to the model is the extracted time expressions, t_i and the model learns to translate it to the normalized time expression, $\hat{t_i}$. Moreover, we also feed the associated sleep event c_i, or the *context* to the model so that the translated result is conditioned based on the context, and hence, we call it *Context Conditioning*. It is often hard to determine the time in the 24-hour clock format, especially when time indicators `AM` or `PM` are not mentioned. For example, in the text *"I woke up at 8"* which results to extracted time expression and sleep event *"8"* and BED_TIME respectively, it is not clear for the model if *"8"* is 8 AM or 8 PM. However, adding the context that *"8"* is associated to the time of going to bed, the model learns that it is

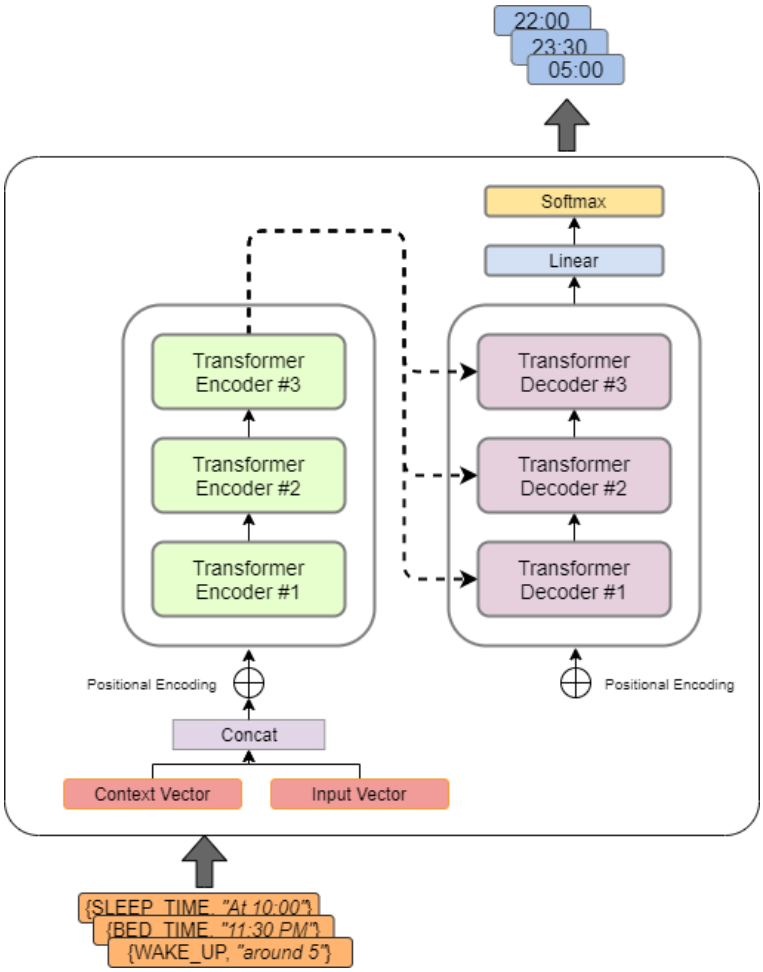

Figure 3.3: Architecture of the adopted Transformer model[100] for Temporal Expression Normalization: We use 3 layers of encoder and decoder each and propose 2 key strategies, *Hybrid Tokenization* and *Context Conditioning*

more probable that time mentioned is in the morning, leading to the result '20:00'. Similarly, *"8 o'clock"* with the context of WAKE_UP helps the model to learn that this is '08:00' in the morning. This serves as our motivation for adding the context as an additional input to the model.

Tokenization

Tokenization plays a significant role in dealing with text data. It is a way of separating a piece of text into smaller units called tokens. Tokens can be either words, characters, or subwords.

For tokenizing the input temporal expressions, we take a hybrid approach. We first tokenize them into word pieces and then further tokenize the words that contain numbers or punctuation into characters. Simply put, words that take the form of temporal prepositions (e.g. on, in or at), temporal conjunctions (e.g. before or after) or prepositions signaling modal-

ity (e.g. to) are represented at word-level and tokens with numerical characters which are essentially the absolute time expressions (for example, 10:00, 9pm, 23.00 and so on) are represented at character-level. Since these time expressions have a probability of having high out-of-vocabulary (OOV) rates, character-level representations are more suitable. They also capture useful orthographic and morphological information.

The sleep events or *context* do not need tokenization since they are one-word entities, and the target normalized time expressions are tokenized as characters for the same reasons as mentioned before. Start-of-sentence and end-of-sentence tags are added at the start and end of tokenized sequence of input temporal expressions, context and tokenized sequence of target normalized temporal expressions.

We take the sequence of tokenized temporal expression $t_i = \{t_{i_1}, t_{i_2}, t_{i_3}, ..., t_{i_{k_i+2}}\}$, where k_i is the number of tokens, and context $c_i = \{c_{i_1}, c_{i_2}, c_{i_3}\}$[2] and concatenate them as,

$$src_i = [t_{i_1}\ t_{i_2}\ t_{i_3}\ ...\ t_{i_{k_i+2}}\ c_{i_1}\ c_{i_2} c_{i_3}] \tag{3.3}$$

The src_i is fed to the encoder which compresses it into a sequence of context vectors $z_i = (z_{i_1}, z_{i_2}, z_{i_3}, ..., z_{i_m})$, where $m = k + 5$ [3]. The tokens are first passed through the embedding layer. Since this model is not recurrent, [100] uses a fixed static embedding to implement the idea about the order of the tokens. However, we implement the idea of the order of tokens by using a second embedding layer called the Positional Embedding Layer as used in modern Transformer architectures like BERT [24]. This is a standard embedding layer where the input is the position of the token within the sequence. The standard and positional embeddings are summed and fed to N encoder layers to obtain the sequence of context vectors z_i.

Each encoder layer comprises of the multi-head attention layer followed by a dropout layer. Multi-head Attention is a novel concept introduced by the original paper. A residual connection is applied and fed to the Layer Normalization layer. The output is passed to a Position-wise feed-forward layer followed by dropout. Another residual connection is then applied, followed by another Layer Normalization layer. The output from each encoder layer is fed to the next one, and parameters are not shared between layers.

The encoded representation z_i is fed to the decoder layers that decodes it into predicted tokens in the target sentence $\hat{t_i}$. The decoder layers are similar to the encoder layers; however, they have two multi-head attention layers instead of one, a *masked multi-head attention layer* over the target sequence and a multi-head attention layer which combines decoder representation and the encoder representation z_i. During training, the target tokens, shifted right, are fed to the standard and positional embedding layer (similar to the encoder) and the embeddings are summed and fed to the decoder made of N decoder layers. The representation from the last decoder layer is fed to a linear layer followed by a softmax layer resulting in the output probabilities. We present a higher level of understanding of the Transformer model. For more details of each layer, please refer to the original paper [100].

[2] t_{i_1} and c_{i_1} are start-of-sentence tags and $t_{i_{k_i+2}}$ and c_{i_3} are end-of-sentence tags

[3] The size of the sequence of context vector is equal to the size of the src vector

At inference, start-of-sentence token is fed to the decoder as the initial input. The decoder outputs the vector of probabilities $y_{i_{m_i}}$ and the token $\hat{t_{i_{m_i}}}$ is given by,

$$\hat{t_{i_{m_i}}} = \text{argmax}\, y_{i_{m_i}} \tag{3.4}$$

where, $y_{i_{m_i}} \in \{0,1\}^d$ and m_i is the the number of tokens in the i^{th} target and d is the length of vocabulary of target. The predicted token is added to the input at every step till the end-of-sentence token is predicted resulting in our translated target sequence $\hat{t_i}$ and inference is stopped.

3.1.3 Time Entity Parsing and Linking

We now have normalized temporal expressions along with their associated sleep event. Before visualizing the timeline, we need to parse the temporal entities in order to get a complete timeline. Specifically, we need to:

- Fill the gaps: Users often do not mention the time of certain sleep events for either brevity or due to forgetfulness. For example, in the following text description, *"I went to bed at 10:00 and fell asleep after 20 minutes. I remember waking up around 2 am to drink some water. I finally woke up at 7 in the morning and got up to get ready."*, the user does not mention when they went back to bed after "2 am", creating a gap in the timeline. If enough information is not provided in the text, these gaps need to be filled by either asking follow up questions, or using heuristics, e.g. adding a SLEEP_TIME event after 5 minutes.
- Deal with *durative* and *relative* types of expressions: When these expressions are encountered, the time duration needs to be added or subtracted from the last or next observed explicit time to determine the absolute value of time.

We also need to deal with the following two types of temporal events that are often observed in the case of disturbed sleep in conjunction with *range* type of expressions:

- Deal with *frequency* type of expressions: If mentioned, sleep_disturbed events are regularly added with the mentioned frequency between the given range of time.
- Deal with *quantitative* type of expressions: If the occurrence of the disturbance is quantified, sleep_disturbed events are added given number of times between the given range of time.

If frequency or quantitative types of expressions are observed in the absence of a range type, sleep_disturbed are created between the time the user first went to sleep and when they finally woke up from their sleep.

After the resolution of these points, we get a complete timeline that is ready to be visualised. We define the periods between two consecutive events as follows:

1. BED_TIME - LIGHT_OFF (or vice-versa): In bed
2. LIGHTS_OFF - SLEEP_TIME or BED_TIME - SLEEP_TIME: Ready for sleep
3. SLEEP_TIME - WAKE_UP or SLEEP_TIME - SLEEP_DISTURBED: Slumber

4. SLEEP_DISTURBED - SLEEP_TIME: Sleep disturbed

5. WAKE_UP - OUT_OF_BED: Awake in bed

These rules are implemented by writing an algorithm that parses the extracted entities and results in a complete timeline. In Appendix C, we present the complete rule-based parsing.

3.2 MLTC-GIN

In this section, we propose a graph interactive-attention network based multi-label text classification model or MLTC-GIN. Our inspiration for this model comes from the MAGnet model [74] that uses graph attention network to model the dependencies or correlations among labels often observed in MLTC tasks, and the LAHA model [39] that proposed the notion of *interactive-attention* to explore the semantic relationship between each document and labels by taking advantage of both document content and label correlation. Based on these two key concepts, we introduce Graph Interaction-Attention Network. In this novel approach, the label features directly interact with the word features of the training documents to calculate interaction attention. This is further used to learn label-to-label correlation by calculating self-attention. We combine the label feature representation from the Graph Interaction-Attention Network with word features extracted from the BERT model to output the predicted labels. The architecture of our MLTC-GIN model is shown in Figure 3.4. We start with a set of training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^D$ are the input features with D dimension, $y_i \in \{0,1\}^C$ is the target vector with C being the number of classes and finally, n is the number of data samples. Target vector y_i has 1 in the j^{th} position if it belongs to the j^{th} class. The MLTC problem can then be modelled as finding the optimal sequence of labels y^* that maximises the conditional probability $p(y|x)$ by,

$$p(y|x) = \prod_{i=1}^{n} p(y_i \mid y_1, y_2, y_3, ..., y_{i-1}, x) \tag{3.5}$$

We employ the BERT model to extract the representation of tokens $X_i \in \mathbb{R}^{s \times r}$ and also the pooled representation of all tokens $X_{i_{pooled}} \in \mathbb{R}^{1 \times r}$ by,

$$X_i,\ X_{i_{pooled}} = f_{BERT}(x_i, \theta_{BERT}) \tag{3.6}$$

where s is the maximum sequence length of the input and r is the dimension of the token embedding.

We introduce the Graph Interaction-Attention Network (GIN), based on the original Graph Attention Network [101] to determine the semantic relation between words and labels in the same latent space and leverage this to capture the dependencies between the labels. GIN takes node features and adjacency matrix that represent the graph data as input. Here, the labels are the nodes in the graph. Since the labels in our data do not have an inherent graph structure, we learn the adjacency matrix during the training phase, hoping that the model will determine the graph, thereby learning the correlation of the labels. We use the pre-trained BERT model to obtain embeddings for each label and use it as our node features. GIN provides us with the attended label features, which are then combined with features extracted by BERT to give the final probability scores.

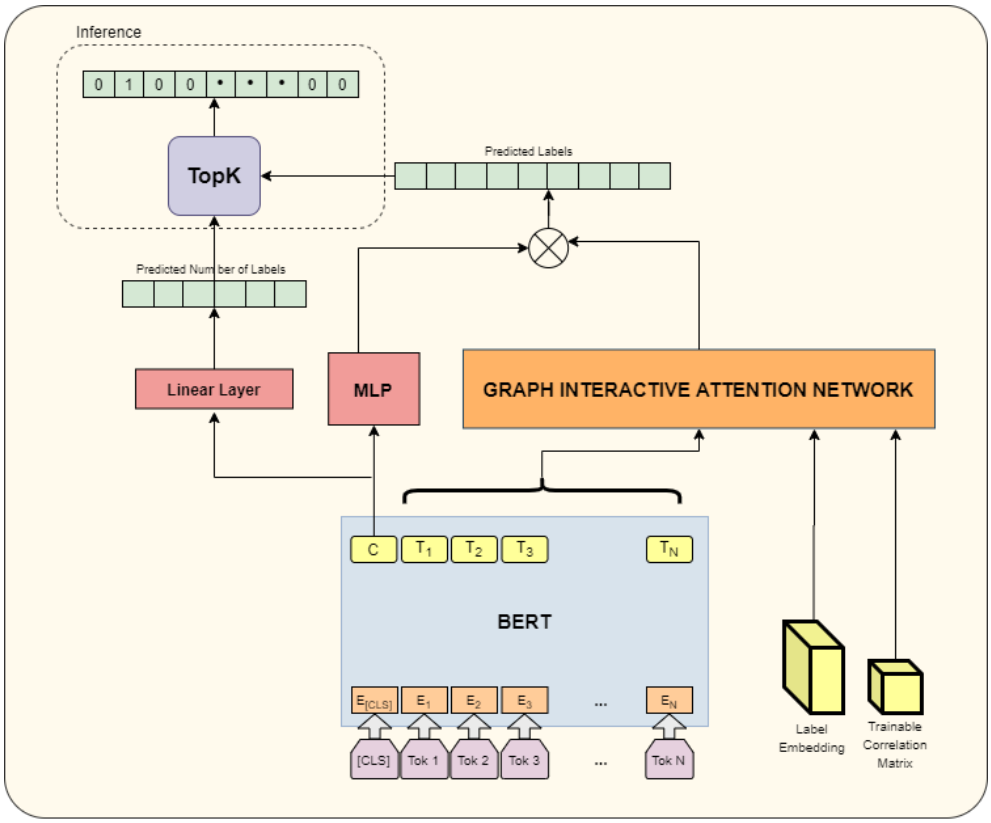

Figure 3.4: Architecture of our proposed model, MLTC-GIN for Multi-label Text Classification tasks using Graph Interaction-Attention Network: Interaction Attention is implemented in a Graph Attention Network to better learn the correlation within the label by also interacting with input features from the text.

3.2.1 Graph Interaction-Attention Network

GIN leverages the attention mechanism to identify label importance in correlation graph by considering the importance of their neighbour labels. We use multi-head attention that uses K different heads. Each layer is independently replicated, meaning they do not share parameters and the outputs are aggregated feature-wise by taking an average,

$$H_i^{l+1} = f(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha_{i,j,k}H_j^l W_q^l) \tag{3.7}$$

where, $\alpha_{i,j}$ is the attention coefficient of label j to label i, $N(i)$ represents the neighborhood of label i in the graph and f is Exponential Linear Unit (ELU) activation function. W_q^l is the bridge mapping matrix (see next section) for the l^{th} GIN layer. A cascade of GIN layers are used in the model and the output from the previous GIN layer is fed into the successive GIN. The first layer takes the label embedding $M \in \mathbb{R}^{C\times r}$ (C is the number of classes and r

is the embedding dimension) as input,

$$H_i^1 = f(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha_{i,j,k}MW_q^0) \tag{3.8}$$

If L number of cascading GIN layers are used, the final attended label features are given as,

$$H_i^L = \underbrace{\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha_{i,j,k}H_i^{L-1}W_q^{L-1}}_{\text{attended label features}} \tag{3.9}$$

Calculating Attention

We propose a new method of calculating the attention coefficients that interact with word representations so that the model not only take advantage of the label correlation but also the document content. As mentioned in the previous section, we use BERT embeddings to represent words and labels in the r-dimensional latent space as X_i and M respectively.

Similar to the work of [39], we align the latent space of words and labels by using a bridge mapping matrix $W_q \in \mathbb{R}^{r\times d}$ and train this matrix with the following objective:

$$Q_L = W_q M \tag{3.10}$$

Key vectors for the word embeddings X_i are calculated by $Q_W = W_x X_i$, where W_x is a trainable weight matrix. We then use the attention queries for each label and key vectors for the words to calculate the interactive matching score $M^{(x)} \in \mathbb{R}^{m\times s}$ as,

$$M^{(x)} = Q_L\ Q_W^T \tag{3.11}$$

We then calculate the label attention with document context $A^{(I)} \in \mathbb{R}^{m\times m}$ from the interactive matching score by using another trainable weight matrix, $W_s \in \mathbb{R}^{s\times m}$,

$$A^{(I)} = M^{(x)}W_s \tag{3.12}$$

We then use softmax to normalise $A^{(I)}$ and get our attention coefficients $\alpha_{i,j}$ as,

$$\alpha_{i,j} = \frac{\exp\left(A_{i,j}^{(I)}\right)}{\sum_{i=1}^{n}\exp\left(A_{i,j}^{(I)}\right)} \tag{3.13}$$

Calculating Final Prediction Score

The pooled output $X_{i_{pooled}}$ from BERT is passed through a Multi-layer perceptron to get the final feature vector F in d dimension. If $H_{gin} \in \mathbb{R}^{m\times d}$ are the attended label features from GIN, then the final prediction score is given by their multiplication,

$$\hat{y} = F \odot H_{gin} \tag{3.14}$$

where, $\odot$ represents the multiplication function and $\hat{y} \in \mathbb{R}^C$.

Loss Function

We use Binary Cross Entropy as the loss function. Given the ground-truth labels as $y \in \{0,1\}^C$ and $\hat{y}$ is the final predicted score, the loss is calculated as:

$$\mathcal{L} = \sum_{c=1}^{C} y^c \log(\sigma(\hat{y}^c)) + (1 - y^c) \log(1 - \sigma(\hat{y}^c)) \tag{3.15}$$

where, σ is the Sigmoid activation function.

3.2.2 Predicting the Number of Labels

The instance-wise prediction of number of labels is modelled as a multi-class classification problem. We use the pooled features from BERT as input and train a linear layer to predict the number of labels for each instance. Given the ground-truth labels as $y_i \in \{0,1\}^C$ and pooled features from the BERT model $X_{i_{pooled}}$, we train a linear layer as:

$$\hat{k} = W_l X_{i_{pooled}} + b \tag{3.16}$$

where, W_l are the weights and b is the bias associated to the linear layer and $\hat{k} \in \mathbb{R}^C$ is the probability score. We take the argmax of the prediction score to find the most probable number of labels for each instance,

$$y_k = \text{argmax}\, \hat{k} \tag{3.17}$$

During evaluation and inference, the Top-K method is used to select the top y_{k_i} probabilities score from the multi-label predictions where y_{ki} is the number of labels predicted for i^{th} instance.

Chapter 4

Experiments & Results

In this chapter, we discuss our experiments and present our results and discussions. We first discuss the various datasets used for training and give detail about their collection and processing in Section 4.1. We then discuss the details of implementation and state the setting of our experiments in Section 4.2. In Section 4.3, we present our results by comparing the performance of the adopted and proposed methods against baselines and state-of-the-art methods. Moreover, we present the results of our ablation studies and simultaneously discuss the outcome. Lastly, we do a qualitative analysis of our proposed pipeline by observing its performance on a sampled set of examples.

4.1 Data

In this section, we describe the datasets used in this thesis and give details about their source and collection. We also describe the pre-processing steps are taken and labelling tools used.

4.1.1 Sleep Narratives Dataset

The Sleep Narratives dataset is a small dataset containing 600 free-text accounts of people's sleep of the previous night. The data was collected by Philips Research using Amazon Mechanical Turk (MTurk). MTurk is a crowdsourcing website for businesses (known as Requesters) to hire remotely located "crowdworkers" to perform discrete on-demand tasks that computers are currently unable to do. The participants were asked to provide information about their sleep of the previous night via an open question, *"Please describe, in a few lines, your sleep last night"*.

Table 4.1: Examples of the type of response expected from the open question *"Please describe, in a few lines, your sleep last night"*

Yesterday, I went to bed at 10:00 and got up at 6 this morning. It took me around 30 minutes to fall asleep. I only woke up once at around 3 o'clock in the morning, due to some noise in the street. I would rate my sleep of this night with an 8.
Last night I slept very badly. I went to bed too late at around half past midnight and could not fall asleep until 3 o'clock in the morning. And at 6 my alarm clock went off. I think I could not fall asleep because of stress at work.

Table 4.2: Examples of the IOB2 or BIO tagging scheme

Tokens	I	went	to	bed	at	10:00	pm	and	got	up	at	6
Tags	O	O	O	O	O	B-BED_TIME	I-BED_TIME	O	O	O	B-WAKE_UP	I-WAKE_UP

The collected free-text data was then pre-processed by removing all accented characters and converting the text to all lower case. NER datasets require the entities in the text to be tagged for the purpose of training. We use the IOB2, or simply BIO, tagging format for tagging temporal expressions in our dataset with their corresponding sleep event class (see Section 3.1 for the list of sleep event classes). This tagging format helps deal with entities that are composed of more than one word. The B-tag, which stands for beginning, is used when the token begins an entity, I-tag, which stand for inside, is used when the token is followed by a token of the same entity, indicating that its inside the entity and finally the O-tag, which stands for outside, is used when the token does not belong to any entity, indicating that its outside the entity. Table 4.2 shows an example of the BIO tagging scheme.

We use Label-Studio[1], an open-source data-labelling tool, to annotate our dataset. The annotations are exported in CoNLL 2003[2] file format which has two columns, one for the tokens of text and the second for corresponding tags in the BIO format. Figure 4.1 shows the document frequency and term frequency of sleep event entities. For our problem, we define the document frequency of a label as the number of data samples that include an entity tagged with that label. The term frequency of a label is defined as the total number of entities that are tagged with that label in the entire dataset. We use 80% as training set, 10% as validation set and 10% as test set.

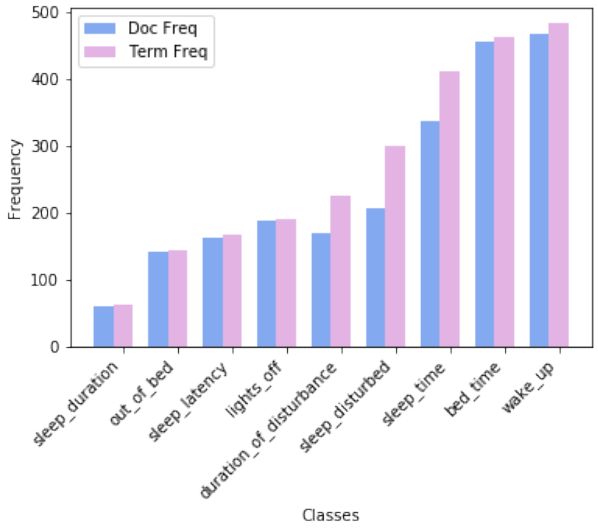

Figure 4.1: Barplot showing the distribution of the Sleep Narratives dataset over the defined classes

[1] https://labelstud.io/
[2] https://www.clips.uantwerpen.be/conll2003/ner/

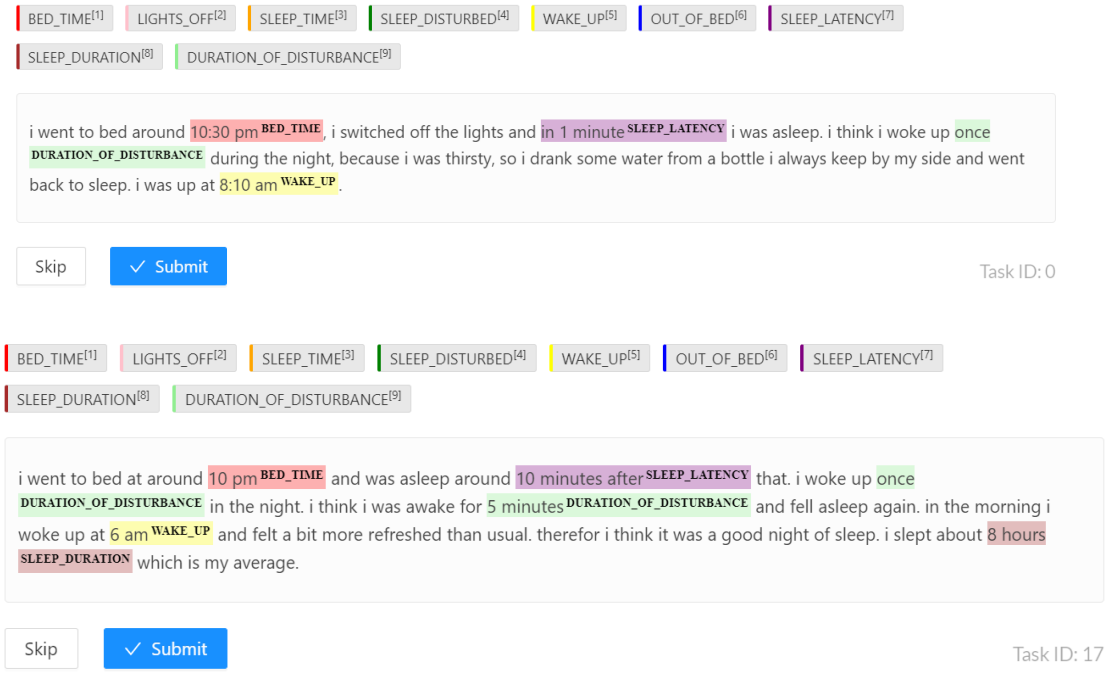

Figure 4.2: Dataset annotation using Label Studio

4.1.2 Time Normalisation Synthetic Dataset

As we have mentioned before, popularly used rule-based time extractors (or taggers) and normalization tools such as HiedelTime and SUTime fail to detect *vague* or *durative* type of time expressions. Taking an example of a simple sleep narrative, *"I went to bed at 10 and fell asleep 10 mins later"*, HiedelTime fails to detect both *"at 10"* and *"after 10 mins"*. Therefore, we adopt the Transformer model that can learn these rules by training while also generalize on newer data. However, we lack access to a dataset that focuses on non-explicit types of temporal expression, and that can enable our model to learn how to normalize them. Therefore. we create a dataset artificially using regular expressions that focuses on the types of expressions we defined in subsection 3.1.2. We define several regular expression patterns for each sleep event and generate a large number of examples for each type using a random number generator. We make sure to generate a random number for each regular expression in a way that leads to the generation of a valid time expression.

For each generated time expression, their normalized equivalent in 24-hour format is generated. When time indicator AM or PM are not generated for the time expressions, we deduce the correct time by using the sleep event class as context. Appendix A presents the different regular expressions created to generate the dataset along with the list of sleep event classes they are used for and the format we use to generate their normalized versions. In total, we generated 30,800 unique triples consisting of the time expression, the sleep event class and normalized time expression. We use 70% as train set, 15% as validation set and 15% as test set.

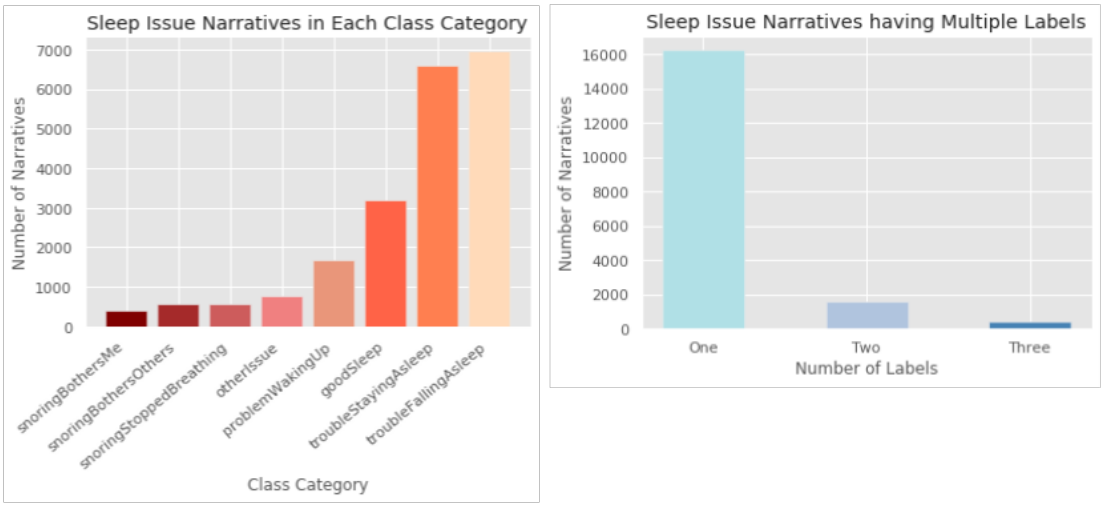

Figure 4.3: Distribution of the Sleep Issues MLTC dataset: Number of data samples per each class category (left) and Number of data samples having multiple labels (right)

4.1.3 Sleep Issues MLTC Dataset

The Sleep Issues dataset is a collection of 18,256 free-text narratives of people describing their sleep-related issues. The data was collected by Philips Research using Amazon Mechanical Turk (MTurk). The participants were asked to imagine that they are having a conversation with a nurse or health-care professional because of some issues they are experiencing related to their sleep. Their narratives were retrieved as an response to the open question , *"Can you tell me a bit about your sleep issue?"*. The collected free-text data was then pre-processed by removing all accented characters and converting the text to all lower case. The processed data was analyzed and manually classified into 8 classes, where any given free-text narrative can belong to more than one class category. Figure 4.3 gives the distribution of the data over 8 class categories and statistics of number of data samples having multiple labels. We use 80% as training set, 10% as validation set and 10% as test set.

4.1.4 Toxic Comment Classification Dataset

We also evaluate the performance of our novel MLTC-GIN model by comparing its performance to state-of-the-art models on a benchmark dataset. We use the toxic comments dataset provided by Kaggle for their Toxic Comments Classification Challenge. The dataset consists of a large number of comments from Wikipedia talk page edits. Human raters have labelled them for toxic behaviour. Eight labels are used to classify a comment, toxic, severe toxic, obscene, threat, insult, identity hate and non-toxic. Due to the limitations in the hardware capacity available to us, we limit the size of our training set to 154,545. However, we use the original test set of size 61,600. We use the same train and test sets for all experiments and use 10% of the train set for validation.

4.2 Implementation Details

We use Python 3.7 as the programming language for all our implementations. All experiments were run five times, and the averaged results have been presented.

We train the BERT+FLAIR BiLSTM-CRF model using the framework [3] developed by [1]. It does away with all embedding-specific engineering complexity and allows researchers to "mix and match" various embeddings with little effort, enabling easy stacking of different embeddings. It also provides support for using BiLSTM-CRF as our sequence tagging model. We stack the FLAIR embeddings pre-trained on 1-billion word corpus [15], and BERT-base uncased embeddings provided by the framework for our implementation. We train the model for 150 epochs with mini-batch size equal to 32. We use SGD optimizer with an annealing learning rate with the initial value set to 0.1 and is reduced with the anneal factor of 0.5 if the training loss does not improve for three epochs. If the minimum learning rate is reached before the complete training, the training is stopped early.

The Transformer model for time normalization was implemented using the PyTorch and torchtext libraries. We train our model for 30 epochs with the early stopping criteria with patience equal to 3. We use Adam optimizer with a fixed learning rate of 0.0005 and use the batch size of 128.

We implement our entity parsing algorithm using pure Python 3.7. We use the Matplotlib library to visualize our sleep timeline. Finally, our proposed pipeline is constructed by using the trained BERT+FLAIR BiLSTM CRF model and Transformer model.

We implement our proposed MLTC-GIN model in PyTorch. We wrap the PyTorch model in an sklearn wrapper for future use of grid search and active learning tools such as ModAL that provide support for sklearn estimators. We train the model for 100 epochs with early stopping criteria with patience 10, meaning the training stops when the chosen metric, F1-score does not improve for 10 epochs.

List of hyper-parameters used for each architecture are presented in Appendix B.

[3] https://github.com/flairNLP/flair

4.3 Results & Discussion

In this section, we present the results of our experiments by doing performance comparisons and performing several ablation studies. Moreover, we qualitatively analyse the performance of our SERN pipeline by running inference on some sampled texts and visualizing the extracted tiemlines.

4.3.1 Performance Comparison

In this section, we present and discuss the results of our experiments and compare the performance of our adopted and proposed algorithms with existing baselines and state-of-the-art methods.

Temporal Expressions Extraction by Sequence Tagging

Table 4.3 shows a comparison of overall F1 scores and CoNLL F1-scores per class category for the Time Extraction NER task using BERT embedding, FLAIR embedding and their combination. The CoNLL F1 score is a strict version of the standard F1 score where a true positive is scored only if all the tokens of a given entity are classified correctly (including their B- and I- tags). Conversely, every incorrect B- prediction is counted as a false positive. When comparing the performance of models that only use either BERT or FLAIR, BERT embedding performs better (↑ 3.36%) indicating that character-level features alone cannot outperform models that learn from word-level features. However, stacking word-level BERT embedding with character-level FLAIR embeddings gives the best performance with an overall F1 score of 66.40, significantly improving over the model that only uses word-level features (↑ 3.68%). This validates our hypothesis that additional use of character-level features as opposed to just using word-level features is particularly helpful for this task.

Focusing on class-wise CoNLL F1-scores, we see that SLEEP_LATENCY, and OUT_OF_BED score lower which can be attributed to the fact that their term and document frequencies are significantly lower when compared to other classes (see Figure 4.1). SLEEP_TIME class also scores low even though their document and term frequencies are high. However, an important detail to note is that we use the class SLEEP_TIME to indicate two types of sleep time: when the person first goes to sleep after going to bed, and when the person goes to sleep after having their sleep disturbed, making it a particularly challenging class and we attribute the lower score to this fact. Comparing the class-wise performance of the three types of embedding, few classes benefit more from either just word-level BERT features or character-level FLAIR features. Temporal entities belonging to class BED_TIME are mostly of *explicit* type which are purely numeric in nature and the absolute time can be extracted without the need for temporal prepositions or conjunctions. This may be a reason why entities from this class benefit from a purely character-level embedding. Temporal entities belong to LIGHTS_OFF, SLEEP_TIME, OUT_OF_BED and SLEEP_DURATION seem to benefit more from only word-level features. Time expressions belonging to these classes often are of non-explicit types using supporting words that are temporal prepositions or conjunctions. This leads us to believe that a better way of integrating the two types of embeddings (instead of simply stacking them) may lead to improvements. Another factor affecting the class-wise behaviour we observe could be inconsistencies in ground-truth annotations across the train and test sets due to human error,

Table 4.3: Performance evaluation of BERT, FLAIR, and BERT+FLAIR based NER architectues on the Sleep Narratives test set: Class-wise CoNLL F1-scores are reported and in the last two rows, overall F1-micro averaged and Accuracy scores are reported for each of the three architectures.

Models	BERT	FLAIR	**FLAIR+BERT**
F1 $_{\text{BED\_TIME}}$	54.55	**74.42**	73.33
F1 $_{\text{LIGHTS\_OFF}}$	**78.57**	56.25	68.97
F1 $_{\text{SLEEP\_TIME}}$	**56.18**	50.55	47.37
F1 $_{\text{SLEEP\_DISTURBED}}$	76.47	61.82	**81.82**
F1 $_{\text{DURATION\_OF\_DISTURBANCE}}$	64.52	56.60	**70.18**
F1 $_{\text{WAKE\_UP}}$	69.81	66.02	**74.29**
F1 $_{\text{OUT\_OF\_BED}}$	**51.85**	42.86	42.86
F1 $_{\text{SLEEP\_LATENCY}}$	37.50	46.81	**47.37**
F1 $_{\text{SLEEP\_DURATION}}$	**85.71**	57.14	75
Overall F1-score	62.72	59.36	**66.40**
Overall Accuracy	95.67	95.59	**96.23**

Temporal Expression Normalization by Neural Machine Translation

To evaluate the performance of our time normalization NMT model, we use BLEU and Exact Match. The Bilingual Evaluation Understudy Score or BLEU [75] for short was developed for quick, inexpensive and language-independent evaluation of automatic machine translation systems that closely relates to human evaluation. Exact Match (EM) is the strictest metric which scores a translation 1 if it is an exact match to the ground truth and 0 otherwise. This metric is often too strict for general NMT tasks e.g. language translation, since there can exist multiple correct translations for a sentence. However, this metric is well suited for our time normalization task since there is always one possible normalized translation for a time expression.

We compare the performance of Transformer model, that we adopt for our pipeline, with some baselines such as, Seq2Seq, RNN Encoder-Decoder, and RNNsearch. Table 4.4 presents the results of five NMT architectures on our time normalization task. RNN encoder-decoder shows significant improvement over Sequence-to-sequence model (BLEU $\uparrow$ 4.01%, EM $\uparrow$ 10.31%), indicating that replacing LSTMs with GRU and relieving information compression leads to improved performance. RNN-search Transformer model performs remarkably well than the other models validating the advantage of using the multi-head attention mechanism.

Multi-Label Text Classification of Sleep Issues

The evaluation measures for multi-label are inherently different than for single-label. In multi-label classification, a misclassification is not entirely wrong or right. A prediction containing a subset of the actual classes should be considered as partially correct and better than a prediction that contains none of them, i.e., predicting two of the three labels correctly this is better than predicting no labels at all. We use multiple metrics to evaluate the performance

Table 4.4: Performance evaluation of several NMT architectures on our Artificial Time Normalization Dataset: BLEU and Exact Match scores are reported for each architecture.

Models	**BLEU-score**	**Exact Match**
Seq2Seq	81.67	38.14
RNN Encoder-Decoder	85.68	48.45
RNN-search	85.35	47.29
Transformer	**98.07**	**97.58**

of our MLTC experiment. Firstly, we use general evaluation metrics such as accuracy and micro-averaged F1 score. We also use Hamming Loss which gives the fraction of the wrong labels to the total number of labels. Since this is a loss function, its optimal value is zero, and its upper bound is one. To incorporate a measure of the ranking quality, we use a rank sensitive metric, Normalized Discounted Cumulative Gain (nDGC), which has been a popular choice of metric for information retrieval tasks and in many multi-label classification tasks as well. This ranking metric yields a high value if true labels are ranked high by the probability estimates. Lastly, we adopt Exact Match (also known as Subset Accuracy) as the strictest measure for our evaluation.

We compare the performance of our proposed method with two kinds of existing architectures: baselines and state-of-the-art. We draw comparisons with baselines such as Binary Relevance (BR) [9], Label Powerset (LP) [99], Classifier Chains (CC) [86], Recurrent Neural Networks (RNN), Recurrent CNN (RCNN) [49] and Hierarchical Attention Networks (HAN) [110]. We implement BR, LP and CC by using `Scikit-Multilearn` [95], an open-source library for the multi-label classification tasks, and use linear-SVM as the base classifier. We use existing implementations of RNN, CNN, CRNN, RCNN and HAN for our experiment[4]. We also compare our proposed method to models that have achieved state-of-the-art performance for various MLTC tasks such as BERT [24], MAGnet [74] and LAHA [39]. Our humble expectation from our proposed method was that it performs significantly better than baseline methods and at least comparable to state-of-the-art methods.

Table 4.5 shows the performance of baseline, state-of-the-art and our proposed methods for the MLTC task on our Sleep Issues dataset. Firstly, our proposed method, MLTC-GIN, outperforms all baseline methods. Results also show that in comparison to state-of-the-art models BERT, MAGnet and LAHA, MLTC-GIN improves on the accuracy ($\uparrow$ 0.08%, $\uparrow$ 0.05%, $\uparrow$ 1.16%), F1-score ($\uparrow$ 0.38%, $\uparrow$ 0.23%, $\uparrow$ 3.56%), Hamming loss ($\downarrow$ 0.0007, $\downarrow$ 0.0004, $\downarrow$ 0.0116%) and nDCG score ($\uparrow$ 0.2%, $\uparrow$ 0.03%, $\uparrow$ 2.41%). On the strictest metric, Exact Match, it outperforms BERT ($\uparrow$ 0.12%) and LAHA ($\uparrow$ 9.61%) models while slightly under performs when compared to MAGnet ($\downarrow$ 0.2%). Figure 4.4 shows the comparison of performance of our proposed model and the state-of-the-art models on the validation set over 60 epochs.

The essential difference in the architecture of MLTC-GIN and MAGnet model is the integration of interactive attention and the use of MLP instead of BiLSTM for feature generation.

[4] `https://github.com/RandolphVI/Multi-Label-Text-Classification`

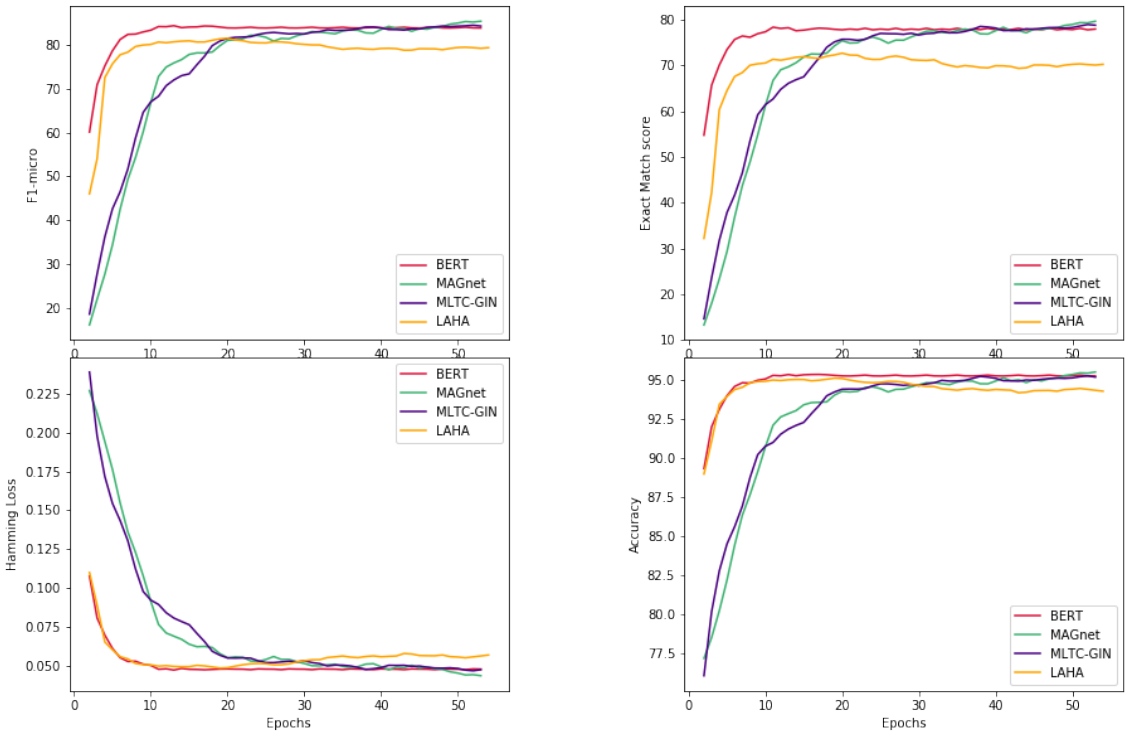

Figure 4.4: Performance of the BERT, MAGnet, LAHA and our proposed model MLTC-GIN on the validation set over 60 epochs on the Sleep Issues dataset.

We can see from the results that adopting these techniques improves the performance on the MLTC dataset. Moreover, the superior performance of MAGnet and MLTC-GIN over the LAHA model can be attributed to the use of GloVe embedding in the latter instead of BERT.

To strengthen our claim, we also compare the performance of MLTC-GIN to the MAGnet on the Toxic Comments Dataset. The results of this experiment are presented in Table 4.6.

Table 4.5: Performance evaluation of various architectures on the Sleep Issues MLTC test set: the first seven rows present performance from baseline architectures, the next three rows present performance of state-of-the-art methods and the last row reports the performance of our proposed methodology. Five evaluation metrics are used, Accuracy, F1-micro, Hamming Loss, Exact Match and Normalized Discounted Cumulative Gains. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" indicates the opposite.

Models	**Accuracy (+)**	**F1-micro (+)**	**HL (-)**	**EM (+)**	**nDCG (+)**
BR	94.68	66.81	0.6681	0.6681	0.6681
LP	94.67	75.25	0.7525	0.7525	0.7525
CC	94.25	73.55	0.6681	0.6681	0.6681
RNN	93.32	75.23	0.0674	0.6989	0.8717
CNN	92.15	73.67	0.0727	0.6670	0.8603
RCNN	94.12	77.87	0.0614	0.7130	0.8842
HAN	93.47	75.28	0.0692	0.6763	0.8690
BERT	95.56	83.98	0.0443	0.7908	0.9241
MAGnet	95.59	84.13	0.0440	**0.7940**	0.9258
LAHA	94.48	80.80	0.0552	0.6959	0.9020
MLTC-GIN	**95.64**	**84.36**	**0.0436**	0.7920	**0.9261**

Table 4.6: Performance evaluation of MAGnet and MLTC-GIN models on the Toxic Comments test set: Five evaluation metrics are used, Accuracy, F1-micro, Hamming Loss, Exact Match and Normalized Discounted Cumulative Gains. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" indicates the opposite.

Models	**Accuracy (+)**	**F1-micro (+)**	**HL (-)**	**EM (+)**	**nDCG (+)**
MAGnet	95.14	87.98	0.0385	87.17	95.19
MLTC-GIN	**96.70**	**89.75**	**0.0329**	**88.74**	**95.62**

4.3.2 Ablation Study

Impact of the CRF Layer

In section 3.1.1, we stated our motivation to use the CRF layer for our time expression extraction model. We perform the NER experiments again on our dataset but replacing the CRF layer with a feed-forward layer. Table 4.7 shows the results of the overall F1 score. Removing the CRF layer from the model architecture results to a decrease in performance for each architecture. This strongly suggests that adding the CRF layer helps the model learn the correlations between the current label and neighbouring labels using the learnt transition matrix.

Impact of Hybrid Tokenkization and Context Conditioning on Time Normalization NMT model

In Section 3.1.2, we discussed our motivation to use a hybrid tokenization scheme for input temporal expressions and the use of their associated sleep event class as an additional input

Table 4.7: Ablation Study of the NER architectures on Sleep Narratives test set to measure the impact of the CRF layer: F1-scores of FLAIR, BERT and FLAIR+BERT based architectures are reported after removing the CRF layer. ↓ indicates the decrease in performance in percentage points.

Models	F1-score
$\text{FLAIR}_{\text{w/o CRF}}$	37.27 (↓ 22.09)
$\text{BERT}_{\text{w/o CRF}}$	60.47 (↓ 2.25)
$\text{FLAIR+BERT}_{\text{w/o CRF}}$	**60.51** (↓ 5.89)

for context. To validate our motivation, we train the Transformer sequence-to-sequence model without the addition of context vector and switching to word-level and character-level tokenizers and analyze their performance. Table 4.8 shows the results of Transformer model on the test set with and without context conditioning using hybrid, word-level and character-level tokenizer for each. Transformer model with context conditioning with hybrid tokenization performs the best with BLEU score of 98.07 and Exact Match score of 97.58. The performance drastically reduces when the hybrid tokenizer is replaced with either a word-level tokenizer (BLEU ↓ 30.39, EM ↓ 58.45) or character-level tokenizer (BLEU ↓ 8.38, EM ↓ 44.74). Performs also drops significantly when context conditioning is not adopted. However, using the hybrid tokenizer still gives the best performance even when context conditioning is not adopted.

Table 4.8: Ablation Study of the NMT architectures on the Artificial Time Normalization test set to measure the impact of adopting hybrid tokenization and Context Condiitoning: BLEU and Exact Match scores of the Transformer model were reported with and without context conditioning with the combination of using hybrid tokenization or replacing it with word-level tokenization and character-level tokenization.

Models	**BLEU score**	**EM**
Transformer $_{\text{+context vector}}$		
w/ hybrid tokenizer	**98.07**	**97.58**
w/ char tokenizer	89.69	52.84
w/ word tokenizer	67.68	39.13
Transformer $_{\text{-context vector}}$		
w/ hybrid tokenizer	88.44	51.05
w/ char tokenizer	87.68	50.92
w/ word tokenizer	64.67	36.04

We further analyze the impact of context conditioning by visualizing the attention weights from the Transformer model when the context vector is used. Figure 4.5 visualizes attention weights as a heat map for two examples from the test set. The relation between the context

Table 4.9: Ablation Study of the proposed MLTC-GIN model against the state-of-the art architectures on Sleep Issues MLTC test set that have more than one ground truth label to analyse the efficacy of the model in learning label correlation: Five evaluation metrics are used, Accuracy, F1-micro, Hamming Loss, Exact Match and Normalized Discounted Cumulative Gains. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" indicates the opposite.

Models	**Accuracy (+)**	**F1-micro (+)**	**HL (-)**	**EM (+)**	**nDCG (+)**
BERT	85.38	68.24	0.1461	20.28	87.12
MAGnet	85.08	68.12	0.1492	20.77	86.83
LAHA	81.96	48.05	0.1803	14.58	78.51
MLTC-GIN	**86.23**	**70.54**	**0.1377**	**23.19**	**88.06**

vector and the hours (`HH`) of the normalized translation can be clearly visualized. In the example, when the context is BED_TIME and the input temporal expression is *"9:30"*, no time indicator such as AM or PM is mentioned, but from the provided context, the model predicts the normalized time as `21:30` i.e., in 24-hour format which is much likely than `09:30` since its a person's bedtime. Attention heads 1, 5, 6, 7, 8 and 9 capture this relation between the context BED_TIME and hour tokens *"2"* and *"1"*.

Inference of Non-Singleton Label Examples

As seen from Figure 4.3, the Sleep Issues MLTC dataset is highly skewed since ≈87% of the text examples only belong to one label, leaving only ≈13% of examples that actually belong to multiple labels. Since results from the single label examples have a large contribution, it is hard to judge if the model actually learns label correlation or if it only improves on predicting single labels correctly. Therefore, we run inference on the subset of the test data that are assigned to more than one label class and see if how well our proposed method has learned to predict multiple labels as compared to the state-of-the-art methods. We present the results in Table 4.9.

Our proposed method MTLC-GIN significantly outperforms the previous state-of-the-art models, including on the strictest evaluation metric Exact Match score. Even with a relatively small fraction of multi-label examples to learn from, MLTC-GIN does a better job at predicting multi-label classes.

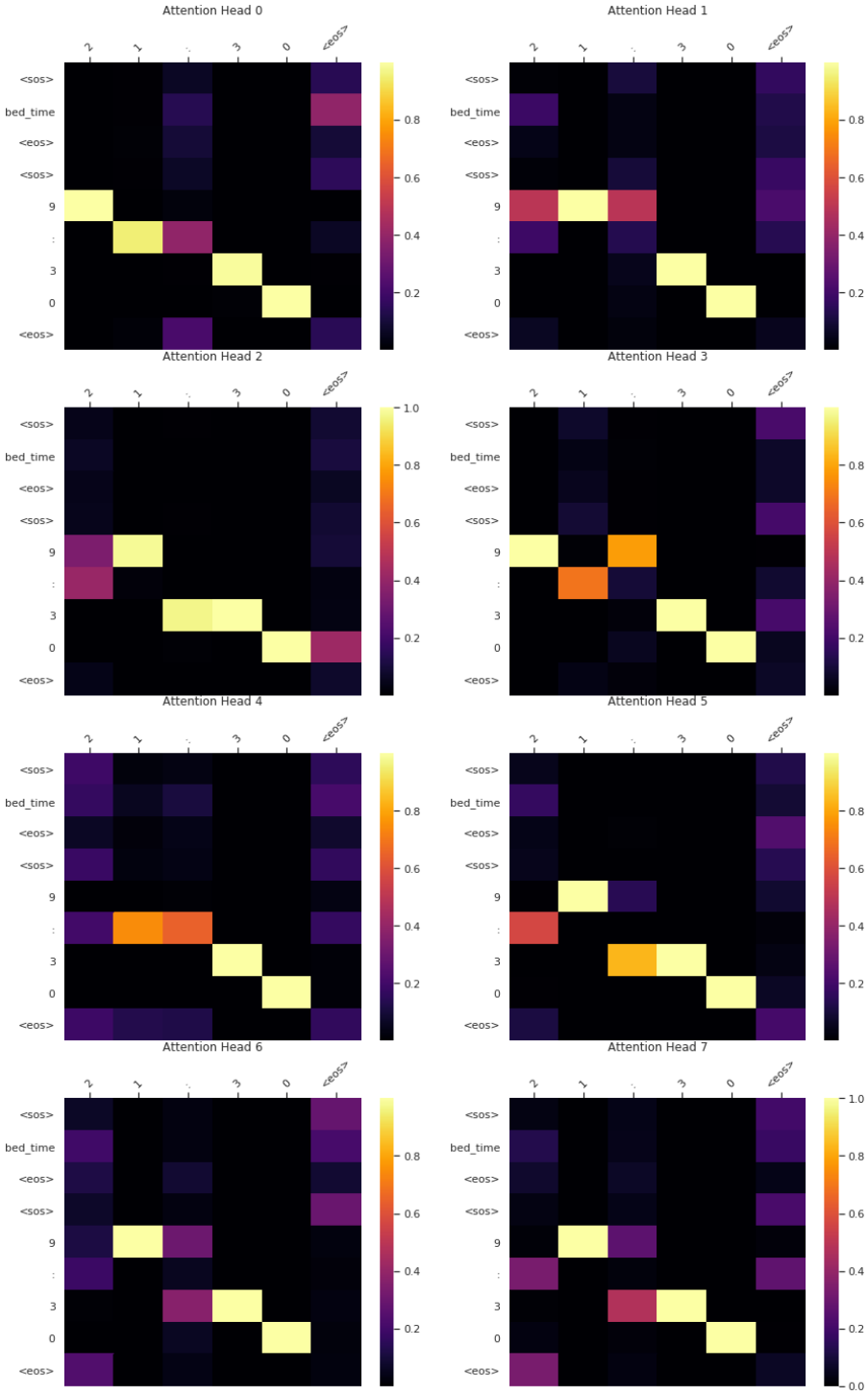

Figure 4.5: Attention weights from 8 attention heads of the Transformer model visualized as heat maps on inference: The temporal expression "9:30" and sleep event class BED_TIME was provided as input and context respectively. The model correctly predicts the translated normalisation as "`21:30`".

4.3.3 Analysis of the SERN Pipeline

We also analyze the results from the Sleep Events Recognition and Normalisation pipeline. It is not feasible to perform a quantitative evaluation on the pipeline since we do not have ground truth timeline extractions to make a comparison with the results from the pipeline. Therefore, we sample some examples from the test data and use the pipeline to extract the timeline. We perform a qualitative analysis on the output and also probe into the intermediate results of the pipeline, i.e., results from the temporal expression extraction and normalization, respectively. Table 4.10 presents the intermediate and final results obtained from the SERN pipeline on 5 sleep narratives randomly sampled from the test set. The overall performance of the pipeline seems satisfactory, and it delivers results that are expected of it. When key sleep events such as SLEEP_TIME and WAKE_UP or BED_TIME in the absence of LIGHT_OFF are missing, the pipeline asked a follow up question to the user to retrieve these time expressions and complete the timeline (e.g. 1^{st}, 2^{nd} and 3^{rd} narratives in Table 4.10). Moreover, it handles *vague* type of time expressions well by using heuristics that seem appropriate (e.g. 1^{st} narrative in Table 4.10). However, we note that the time expression extraction component that uses a BERT and FLAIR embeddings based NER model is the bottleneck of our pipeline. In the 5^{th} narrative, e.g., the time expression *"45 minutes"* from the phrase *"it took me 45 minutes to fall back asleep"* is mislabeled as SLEEP_LATENCY instead of SLEEP_TIME since these events are quite similar with the difference being that a *durative* SLEEP_TIME is observed after disturbance in the sleep and SLEEP_LATENCY is observed after bedtime or time when the lights are turned off. This error is further propagated in the pipeline. Since the presence of sleep_latency after disturbed sleep is not possible and therefore not expected by the parsing algorithm, this breaks the cycle of sleep disturbance as the algorithm assumes that there are no further disturbances since the next time event is neither SLEEP_DISTURBED, DURATION_OF_DISTURBANCE or SLEEP_TIME. As a result, the next disturbed time i.e., "4:30" is skipped.

Figure 4.6 shows the visualizations of the extracted timelines presented in Table 4.10. We plot the span between two sleep events as bars and use lollipop graphs to visualize the time of the sleep event with the *x-axis* centered around midnight (`00:00`).

Table 4.10: Final and intermediate results from the SERN pipeline upon inference on 5 randomly samples test examples: The first column presents the processed sleep narratives and the subsequent rows present the extracted time expression and associated sleep event class, the normalised time expression and finally the parsed completed timeline respectively.

Sleep Narrative	**Extracted Temporal Entities**		**Normalised Entities**		**Extracted Timeline**	
	Sleep Event	Extracted Timex	Sleep Event	Norm Timex	Sleep Event	Timex
i went to sleep about 11 p.m. the room was quiet and dark and the temperature was comfortable. it took me a few minutes to get to sleep as i was thinking about the plot in a program i had been watching. the next thing i remember is waking up this morning when the alarm went off, feeling somewhat groggy but refreshed.	BED_TIME	*11 p.m.*	BED_TIME	23:00	BED_TIME	23:00
	SLEEP_TIME	*a few minutes*	SLEEP_TIME	+00:05	SLEEP_TIME	23:05
			WAKE_UP[6]	8:00	WAKE_UP	08:00
i went to bed at about 9:30. stayed up until about 11, which is late for me, watching the basketball game. the lights went out shortly after that. i fell asleep soon after taking the tylenol. after that i slpet good until i wok up at 5:45. i felt fairly refreshed after waking up.	BED_TIME	*about 9:30.*	BED_TIME	21:30	BED_TIME	21:30
	WAKE_UP	*5:45*	SLEEP_TIME[6]	23:15	SLEEP_TIME	23:15
			WAKE_UP	05:45	WAKE_UP	05:45
i went to bed around 12 am. this morning. i did not switch off the lights as i fell asleep on the couch watching a movie. i went to sleep fairly quickly. i woke up this morning around 6:30 am. overall, i got about 6 hours of sleep and feel very good this morning.	BED_TIME	*12 am*	BED_TIME	00:00	BED_TIME	00:00
	WAKE_UP	*6:30 am*	SLEEP_TIME[6]	00:20	SLEEP_TIME	00:20
	SLEEP_DURATION	*6 hours*	WAKE_UP	06:30	WAKE_UP	06:30
i turned the lights off at 9:30 p.m. and fell asleep at approximately 11:10 p.m. i woke up for the first time at 2:30 a.m. and used the bathroom. i was quite groggy. i lied back down and took approximately 45 minutes to fall back asleep. i then woke up again at 4:30, felt slightly better, and fell back asleep at 5:00 a.m. i woke up for good around 5:50 a.m. and lied in bed until 6:10 a.m. overall, it was a struggle to get out of bed, as my sleep was not very restful.	LIGHTS_OFF	*9:30 p.m.*	LIGHTS_OFF	21:30	LIGHTS_OFF	21:30
	SLEEP_TIME	*11:10 p.m.*	SLEEP_TIME	23:10	SLEEP_TIME	23:10
	SLEEP_DISTURBED	*2:30 a.m.*	SLEEP_DISTURBED	02:30	SLEEP_DISTURBED	02:30
	SLEEP_LATENCY	*45 minutes*	SLEEP_LATENCY	+00:45	SLEEP_TIME	02:35
	SLEEP_DISTURBED	*4:30,*	SLEEP_DISTURBED	04:30	WAKE_UP	05:50
	SLEEP_TIME	*5:00 a.m.*	SLEEP_TIME	05:00	OUT_OF_BED	06:10
	WAKE_UP	*5:50 a.m.*	WAKE_UP	05:50		
	OUT_OF_BED	*6:10 a.m.*	OUT_OF_BED	06:10		
i was in the bed at 9 o'clock. normally it takes me 2 - 3 hours to get to sleep. but last night i went to sleep around 9:30. it was raining last night so that kind of soothed me to fall asleep a little quicker than normal. i got up once at 12:30 to get something to drink then i feel back asleep 1 a.m. then i woke up at 5:10 am	BED_TIME	*9 o'clock.*	BED_TIME	21:00	BED_TIME	21:00
	SLEEP_LATENCY	*2 - 3 hours*	SLEEP_LATENCY	+02:03	SLEEP_TIME	21:30
	SLEEP_TIME	*9:30*	SLEEP_TIME	21:30	SLEEP_DISTURBED	00:30
	SLEEP_DISTURBED	*12:30*	SLEEP_DISTURBED	00:30	SLEEP_TIME	01:00
	SLEEP_TIME	*1 a.m.*	SLEEP_TIME	01:00	WAKE_UP	05:10
	WAKE_UP	*5:10 am*	WAKE_UP	05:10		

Figure 4.6: Visualizations of the timelines extracted in Table 4.10

Chapter 5

Conclusions

We present two main contributions for information extraction from free-text sleep narratives that can be adopted for digital sleep monitoring. We first introduced a novel pipeline SERN for Sleep Events Recognition and Normalization. The SERN pipeline takes in free-text sleep narrative of an individual's previous night's sleep and extracts a complete and structured timeline that can be visualized for sleep tracking. The pipeline does this in three stages. We first train a NER model that stacks state-of-the-art word-level BERT embeddings and character-level FLAIR embeddings to extract features from the text and then use a BiLSTM-CRF architecture for sequence tagging. The model detects temporal entities in the text and tags them with their associated sleep event class. Our experiments showed that using BERT and FLAIR together gave a better overall performance as compared to using either one of them alone. To normalize the ambiguity that comes with natural human language, we train the Transformer model for temporal expression normalization. We propose two strategies, Hybrid Tokenization and Context Conditioning that improves on the performance of the model. We also write a rule-based parsing algorithm that parses the extracted and normalized temporal entities to fill in any gaps in the timeline and handle non-explicit types of temporal expressions. We then visualize the sleep timeline from the extracted structured timeline.

Sleep tracking alone is not beneficial for digital sleep monitoring, and improvement and detecting of issues faced by individuals is essential. Hence, for our second contribution, we propose a novel architecture called MLTC-GIN or Multi-label Text Classification using Graph Interactive-Attention Network that aims at leveraging the relations between the words of the text and labels to capture the dependencies between the labels. Our experiments revealed that MLTC-GIN outperforms baseline architectures and state-of-the-art models on our Sleep Issues dataset and outperforms the current state-of-the-art on benchmark Toxic Comments dataset. We do further experiments on only multi-label examples and prove that our proposed method is better at learning the dependencies within the labels.

5.1 Limitations

The methods presented in this thesis suffer from some limitations as well. The biggest limitation of our proposed SERN pipeline is the absence of an end-to-end evaluation of the pipeline since that would require a dataset that has a structured timeline as ground truth values which we lack. Moreover, our adopted choice of NER architecture, BERT+FLAIR does not perform better than its standalone counterparts on some class labels. This could be attributed to our next limitation, which is a low number of training examples and possible inconsistencies in ground truth annotations due to human error. Lastly, our parsing algorithm finds it hard to construct an accurate structured timeline when sleep disturbed patterns are very intricate and involve multiple *non-explicit* type of temporal expressions. The need for a parsing algorithm comes from the fact that we normalize time expression individually instead of in one pass with the expressions passed as a sequence. However, we are unable to do that since learning such a mapping requires curation of a dataset that can enable such a mapping which was not a possibility for this thesis due to time constraints.

Although our proposed MLTC-GIN model outperforms the current state-of-the-art on our Sleep Issues dataset and the Toxic Comments dataset by Kaggle, its performance is yet to be evaluated on datasets with a large number of labels. Datasets with a larger number of labels result in the training of a larger correlation matrix and hence is out of our scope due to our limited hardware capacity.

5.2 Future Work

Several possible directions can be taken for future work. First, new techniques to integrate the BERT and FLAIR embeddings could be adopted instead of simple concatenation which may result in better feature representation. For our work, we use linear annotations and format to label and normalize our NER data, respectively. However, in the future, hierarchical annotation and compositional structures for normalization, like the SCATE scheme by [8] can be used of the temporal expressions which can help in dealing with complex time concepts. We believe adopting a more intricate normalization structure would also result in the formation of more accurate rules for the parsing algorithm.

For our multi-label text classification task, we train on a dataset that is highly skewed since a large percent of the samples belong to only one label. In the future, data augmentation methods can be adopted to balance the dataset and possibly improve on performance. Moreover, the efficacy of MLTC-GIN should also be tested on a dataset with larger labels. We believe that the exploration of an effective solution to elevate the difficulty of training larger correlation matrices is a critical path to take in the future. Knowing that correlation matrices generally have sparse connections, adopting Sparse Neural Networks could be an interesting approach.

Bibliography

[1] N. Afzal, V. P. Mallipeddi, S. Sohn, H. Liu, R. Chaudhry, C. Scott, I. Kullo, and Adelaide M. Arruda-Olson. Natural language processing of clinical notes for identification of critical limb ischemia. *International journal of medical informatics*, 111:83–89, 2018. 10, 31

[2] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. v, v, 9, 11, 17, 18

[3] Reem Al-Otaibi, Peter Flach, and Meelis Kull. Multi-label classification: A comparative study on threshold selection methods. 09 2014. 14

[4] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 13

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. 7

[6] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press, 2001. 6

[7] Steven Bethard. Cleartk-timeml: A minimalist approach to tempeval 2013. In *SemEval@NAACL-HLT*, 2013. 11

[8] Steven Bethard and Jonathan Parker. A semantically compositional annotation scheme for time normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3779–3786, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). 44

[9] Matthew Boutell, Jiebo Luo, Xipeng Shen, and Christopher Brown. Learning multi-label scene classification. *Pattern Recognition*, 37:1757–1771, 09 2004. 12, 34

[10] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks, 2016. 7

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya

Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 9

[12] Angel Chang and Christoper Manning. Sutime: A library for recognizing and normalizing time expressions. 11 2012. 11

[13] Nai-Wen Chang, Hongjie Dai, Jitendra Jonnagaddala, Chih-Wei Chen, Richard Tzong-Han Tsai, and Wen-Lian Hsu. A context-aware approach for progression tracking of medical concepts in electronic medical records. *Journal of Biomedical Informatics*, 58S, 09 2015. 10

[14] Brian Chapman, Sean Lee, Hyunseok Kang, and Wendy Chapman. Document-level classification of ct pulmonary angiography reports based on an extension of the context algorithm. *Journal of biomedical informatics*, 44:728–37, 03 2011. 10

[15] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. 31

[16] David Chen, Robert Kowatch, Simon Lin, M Splaingard, and Y Huang. Interactive cohort identification of sleep disorder patients using natural language processing and i2b2. *Appl Clin Inform*, 6:345–363, 07 2015. 10

[17] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383, 2017. 13

[18] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. 7

[19] Eun Kyoung Choe, Sunny Consolvo, Nathaniel F. Watson, and Julie A. Kientz. Opportunities for computing technologies to support healthy sleep behaviors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 3053–3062, New York, NY, USA, 2011. Association for Computing Machinery. 2

[20] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. Gram: Graph-based attention model for healthcare representation learning, 2017. 10

[21] Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Yuan Xue, and Andrew Dai. Graph convolutional transformer: Learning the graphical structure of electronic health records, 06 2019. 10

[22] Amanda Clare and Ross D King. *Knowledge Discovery in Multi-label Phenotype Data*, pages 42–53. Lecture Notes in Computer Science. Springer Nature, United States, 2001. 13

[23] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc., 2015. 9

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. v, 9, 13, 17, 18, 21, 34

[25] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China, July 2015. Association for Computational Linguistics. 7

[26] Rosa Figueroa and Christopher Flores. Extracting information from electronic medical records to identify obesity status of a patient based on comorbidities and bodyweight measures. pages 37–46, 12 2015. 10

[27] Samah Jamal Fodeh, Dezon Finch, Lina Bouayad, Stephen L Luther, Han Ling, Robert D Kerns, and Cynthia Brandt. Classifying clinical notes with pain assessment using machine learning. *Medical amp; biological engineering amp; computing*, 56(7):1285—1292, July 2018. 10

[28] Carol Friedman, Lyudmila Shagina, Yves Lussier, and George Hripcsak. Automated Encoding of Clinical Documents Based on Natural Language Processing. *Journal of the American Medical Informatics Association*, 11(5):392–402, 09 2004. 10

[29] J. Fürnkranz, E. Hüllermeier, Eneldo Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73:133–153, 2008. 12

[30] Marzyeh Ghassemi, Tristan Naumann, Finale Doshi-Velez, Nicole Brimmer, Rohit Joshi, Anna Rumshisky, and Peter Szolovits. Unfolding physiological state: Mortality modelling in intensive care units. *KDD : proceedings. International Conference on Knowledge Discovery amp; Data Mining*, 2014:75—84, August 2014. 10

[31] S. Gopal and Y. Yang. Hierarchical bayesian inference and recursive regularization for large-scale classification. *ACM Trans. Knowl. Discov. Data*, 9:18:1–18:23, 2015. 13

[32] S. Gopal, Y. Yang, B. Bai, and Alexandru Niculescu-Mizil. Bayesian models for large-scale hierarchical classification. 3:2411–2419, 01 2012. 13

[33] Siddharth Gopal and Yiming Yang. Multilabel classification with meta-level features. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, page 315–322, New York, NY, USA, 2010. Association for Computing Machinery. 12

[34] A. Graves, Navdeep Jaitly, and Abdel rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013. 7

[35] Alex Graves. Generating sequences with recurrent neural networks, 2014. 7

[36] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009. 10

[37] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018. 9

[38] Kung-Hsiang Huang, Mu Yang, and Nanyun Peng. Biomedical event extraction with hierarchical knowledge graphs, 2020. 12

[39] Xin Huang, Boli Chen, Lin Xiao, and Liping Jing. Label-aware document representation via hybrid attention for extreme multi-label text classification, 2019. 13, 23, 25, 34

[40] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging, 2015. 11

[41] Aiwen Jiang, Chunheng Wang, and Yuanping Zhu. Calibrated Rank-SVM for multi-label image categorization. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*, pages 1450–1455, Hong Kong, 2008. IEEE. 13

[42] Hyuckchul Jung and Amanda Stent. ATT1: Temporal annotation using big windows and rich syntactic and semantic features. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 20–24, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. 11

[43] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences, 2014. 7

[44] Suranga N. Kasthurirathne, Brian E. Dixon, Judy Gichoya, Huiping Xu, Yuni Xia, Burke Mamlin, and Shaun J. Grannis. Toward better public health reporting using existing off the shelf approaches: The value of medical dictionaries in automated cancer detection using plaintext medical data. *Journal of Biomedical Informatics*, 69:160 – 176, 2017. 10

[45] Jonas Kemp, Alvin Rajkomar, and Andrew M. Dai. Improved hierarchical patient classification with language model pretraining over clinical notes, 2019. 10

[46] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014. 7, 13

[47] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models, 2015. 7

[48] Reinhard Kneser and H. Ney. Improved backing-off for m-gram language modeling. *1995 International Conference on Acoustics, Speech, and Signal Processing*, 1:181–184 vol.1, 1995. 6

[49] Siwei Lai, L. Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015. 13, 34

[50] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. 9

[51] Robert Leaman, Ritu Khare, and Zhiyong lu. Challenges in clinical natural language processing for automated disorder normalization. *Journal of biomedical informatics*, 57, 07 2015. 10

[52] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019. 10

[53] Zilu Liang and Bernd Ploderer. Sleep tracking in the real world: a qualitative study into barriers for improving sleep. In *OzCHI '16*, 2016. 2

[54] Bill Y. Lin, Frank Xu, Zhiyi Luo, and Kenny Zhu. Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. 11

[55] Monica Liu and Jiun-Hung Chen. A multi-label classification based approach for sentiment classification. *Expert Systems with Applications*, 42:1083–1093, 02 2015. 12

[56] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. 9

[57] Hector Llorens, Leon Derczynski, R. Gaizauskas, and Estela Saquete Boró. Timen: An open temporal expression normalisation resource. In *LREC*, 2012. 12

[58] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. 8

[59] Ling Luo, Zhihao Yang, Pei Yang, Yin Zhang, Lei Wang, Hongfei Lin, and Jian Wang. An attention-based bilstm-crf approach to document-level chemical named entity recognition. *Bioinformatics (Oxford, England)*, 34, 11 2017. 11

[60] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. 7

[61] Inderjeet Mani and George Wilson. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, page 69–76, USA, 2000. Association for Computational Linguistics. 12

[62] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. 6

[63] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. volume 2, pages 1045–1048, 01 2010. 7

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013. 6

[65] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006. 13

[66] Riccardo Miotto, Li Li, and Brian Kidd. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6:26094, 05 2016. 10

[67] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 10

[68] Chinmoy Nath, Mazen Albaghdadi, and Siddhartha Jonnalagadda. A natural language processing tool for large-scale data extraction from echocardiography reports. *PloS one*, 11:e0153749, 04 2016. 10

[69] Phuoc Nguyen, Truyen Tran, Nilmini Wickramasinghe, and Svetha Venkatesh. Deepr: A convolutional net for medical records, 2016. 10

[70] Yizhao Ni, Jordan Wright, John Perentesis, Todd Lingren, Louise Deleger, Megan Kaiser, Isaac Kohane, and Imre Solti. Increasing the efficiency of trial-patient matching: Automated clinical trial eligibility pre-screening for pediatric oncology patients clinical decision-making, knowledge support systems, and theory. *BMC medical informatics and decision making*, 15:28, 04 2015. 10

[71] S Nowakowski, J Razjouyan, A D Naik, R Agrawal, K Velamuri, S Singh, and A Sharafkhaneh. 1180 the use of natural language processing to extract data from psg sleep study reports using national vha electronic medical record data. *Sleep*, $43(\text{Supplement}_1) : A450 - -A451, 052020.$ 10

[72] Eric Okafor and Charles Osuagwu. The underlying issues in knowledge elicitation. *Interdisciplinary Journal of Information, Knowledge, and Management*, 1, 01 2006. 10

[73] John Oller. On the relation between syntax, semantics and pragmatics. *Linguistics*, 10:43–55, 09 1972. 1

[74] Ankit Pal, Selvakumar Murugan, and Malaikannan Sankarasubbu. Multi-label text classification using attention-based graph neural network, 03 2020. 13, 23, 34

[75] Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. Bleu: a method for automatic evaluation of machine translation. 10 2002. 33

[76] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Bao Mengjiao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. pages 1063–1072, 04 2018. 13

[77] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. 7

[78] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018. 9

[79] Hanieh Poostchi, Ehsan Zare Borzeshi, and Massimo Piccardi. BiLSTM-CRF for Persian named-entity recognition ArmanPersoNERCorpus: the first entity-annotated Persian dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). 11

[80] J. Pustejovsky, José M. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, G. Katz, and Dragomir R. Radev. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering*, 2003. 11

[81] A. Radford. Improving language understanding by generative pre-training. 2018. 9

[82] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 9

[83] Arun Rai. Explainable ai: from black box to glass box. *Journal of the Academy of Marketing Science*, 48, 12 2019. 8

[84] Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. Unsupervised pretraining for sequence to sequence learning, 2018. 9

[85] Jesse Read. A pruned problem transformation method for multi-label classification. In *In: Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS*, pages 143–150, 2008. 13

[86] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. volume 85, pages 254–269, 08 2009. 13, 34

[87] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016. 8

[88] S. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16:235–240, 1994. 13

[89] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. 9

[90] Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 09 2010. 10

[91] Irena Spasić, Jacqueline Livsey, John A. Keane, and Goran Nenadić. Text mining of cancer-related information: Review of current status and future directions. *International Journal of Medical Informatics*, 83(9):605 – 623, 2014. 10

[92] Jannik Strötgen, Julian Zell, and Michael Gertz. HeidelTime: Tuning English and developing Spanish resources for TempEval-3. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 15–19, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. 11

[93] Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813, 04 2013. 11

[94] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. 7

[95] Piotr Szymański and Tomasz Kajdanowicz. A scikit-based python environment for performing multi-label classification, 2018. 34

[96] Lei Tang, Suju Rajan, and Vijay Narayanan. Large scale multi-label classification via metalabeler. pages 211–220, 01 2009. 14

[97] Zhe Tian, Simon Sun, Tewodros Eguale, and Christian Rochefort. Automated extraction of vte events from narrative radiology reports in electronic health records: A validation study. *Medical care*, Publish Ahead of Print, 04 2015. 10

[98] Eugene Tseytlin, Kevin Mitchell, Elizabeth Legowski, Julia Corrigan, Girish Chavan, and Rebecca Jacobson. Noble – flexible concept recognition for large-scale biomedical natural language processing. *BMC Bioinformatics*, 17, 12 2016. 10

[99] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09 2009. 12, 34

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. v, v, 8, 19, 20, 21

[101] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. 13, 23

[102] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator, 2015. 7

[103] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. 17

[104] Jessica A. Walsh, Yijun Shao, Jianwei Leng, Tao He, Chia-Chen Teng, Doug Redd, Qing Treitler Zeng, Zachary Burningham, Daniel O. Clegg, and Brian C. Sauer. Identifying axial spondyloarthritis in electronic medical records of us veterans. *Arthritis Care & Research*, 69(9):1414–1420, 2017. 10

[105] Wei Wang, Kory Kreimeyer, Emily Jane Woo, Robert Ball, Matthew Foster, Abhishek Pandey, John Scott, and Taxiarchis Botsis. A new algorithmic approach for the extraction of temporal associations from clinical narratives with an application to medical product safety surveillance reports. *Journal of Biomedical Informatics*, 62:78 – 89, 2016. 10

[106] Yue Wang, Jin Luo, Shiying Hao, Haihua Xu, Andrew Young Shin, Bo Jin, Rui Liu, Xiaohong Deng, Lijuan Wang, Le Zheng, Yifan Zhao, Chunqing Zhu, Zhongkai Hu, Changlin Fu, Yanpeng Hao, Yingzhen Zhao, Yunliang Jiang, Dorothy Dai, Devore S. Culver, Shaun T. Alfreds, Rogow Todd, Frank Stearns, Karl G. Sylvester, Eric Widen, and Xuefeng B. Ling. Nlp based congestive heart failure case finding: A prospective analysis on statewide electronic medical records. *International Journal of Medical Informatics*, 84(12):1039 – 1047, 2015. 10

[107] Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. *OntoNotes: A Large Training Corpus for Enhanced Processing*. 01 2011. 11

[108] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. Sgm: Sequence generation model for multi-label classification, 2018. 13

[109] Yiming Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 137–145, New York, NY, USA, 2001. Association for Computing Machinery. 14

[110] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics. 13, 34

[111] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038 – 2048, 2007. 13

[112] Yuhui Zhang, Allen Nie, and James Zou. Large-scale generative modeling to improve automated veterinary disease coding, 2018. 10

[113] Dengyong Zhou, Lin Xiao, and Mingrui Wu. Hierarchical classification via orthogonal transfer. pages 801–808, 01 2011. 13

[114] Li Zhou, Yifan Lu, Christopher Vitale, Perry Mar, F Chang, Neil Dhopeshwarkar, and R.A. Rocha. Family relatives as structured data in electronic health records. *Applied clinical informatics*, 5:349–67, 07 2014. 10

Appendix A

Data Creation for Time Normalization

Table A.1: Regular expression patterns used to generate synthetic data for training the Transformer model to translate time expressions to normalised time expressions: The first column lists the regular expression patterns used to generate different types of temporal expressions, in the second column we list the sleep event classes that the generated data would be classified in and the next column presents the normalization format adopted for each type.

Regex Expression	**Sleep Event**	**Normalization Format**	**Example**	
			Timex	**Normalized**
`[0-1]?[0-9][:,.][0-5][0-9] ?((a.?m.?)|(p.?m.?)|(hrs?)|(hours?))?`	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_DISTURBED WAKE_UP OUT_OF_BED	`HH:MM`	*9.00 am*	`21:00`
`[0-1]?[0-9] o'clock`	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_DISTURBED WAKE_UP OUT_OF_BED	`HH:MM`	*8 o'clock*	`22:00`
`((at)|(around)|(until)|(by)) [0-1]?[0-9]`	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_DISTURBED WAKE_UP OUT_OF_BED	`HH:MM`	*at 9*	`21:00`
`((quarter past)|((half|halve) past)|(quarter to)) [0-1]?[0-9]`	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_DISTURBED WAKE_UP OUT_OF_BED	`HH:MM`	*half past nine*	`21:30`

Regex Expression	**Sleep Event**	**Normalization Format**	**Example**	
			Timex	**Normalized**
((a couple of)|(a few)|(several)) ((minutes?)|(mins?)|(hours?)|(hrs?)|) ?((after)|(later)|(after)|(before))?	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_LATENCY SLEEP_DISTURBED WAKE_UP OUT_OF_BED	+HH:MM	*a few mins later*	+00:05
[0-9][0-9]?-?([0-9][0-9]?)? ?((hrs?)|(hours?)|(minutes?)|(mins?))	SLEEP_DURATION	+HH:MM	*2-3 hours*	+02:00
((after)|(within)|(before))? [0-9][0-9]? ?((((hrs?)|(hours?)) (and a half)?)|(minutes?)|(mins?)) ((after)|(later)|(before))	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_LATENCY SLEEP_DISTURBED WAKE_UP OUT_OF_BED	+HH:MM	*2 hours before*	-02:00
(around)?[0-9][0-9]? ?((((hrs?)|(hours?))(and a half)?)|(minutes?)|(mins?)) ((after)|(later)|(before))?	BED_TIME LIGHTS_OFF SLEEP_TIME SLEEP_LATENCY SLEEP_DISTURBED WAKE_UP OUT_OF_BED	+HH:MM	*after 40 mins*	+00:40
[0-9]*-?[0-9]* times?	DURATION_OF_ DISTURBANCE	Tn	5-*6 times*	T5
((several)|(a lot of)|(many)|(couple of)) times?	DURATION_OF_ DISTURBANCE	Tn	*a lot of times*	T3
every [0-9]*-?[0-9]*? ((minutes?)|(mins?)|(hours?)|(hrs?))	DURATION_OF_ DISTURBANCE	*HH:MM	*every 20 minutes*	*00:20
([0-1]?[0-9][:,.]?([0-5][0-9])?) texttt?((to)—-) ?([0-1]?[0-9][:,.]? ([0-5][0-9])?) ?((a.?m.?)|(p.?m.?))	DURATION_OF_ DISTURBANCE	HH:MM-HH:MM	*2:30 to 3:30*	02:30-03:30

Appendix B

List of Hyper-parameters

Table B.1: List of hyper-paramters for the FLAIR+BERT+BiLSTM-CRF NER model

Vocab Size	**Embedding Size**			**BiLSTM Hidden Size**
	FLAIR	**BERT**	**Stacked (FLAIR+BERT)**	
28	4096	768	4196	256

Table B.2: List of hyper-parameters for the Transformer model for Time Normalisation

Encoder			
Hidden Dim	**# of Layers**	**# of Attention Heads**	**Positional Embedding Dim**
256	3	8	512
Decoder			
Hidden Dim	**# of Layers**	**# of Attention Heads**	**Positional Embedding Dim**
256	3	8	512

Table B.3: List of hyper-parameters for the MLTC-GIN model

Vocab Size	**Embedding Size**	**Hidden Dim**	**# of Attention Heads**	**# of GIN layers**
30522	768	200	8	2

Appendix C

Entity Parsing Algorithm

This chapter presents the Entity Parsing algorithm used for structuring the extracted temporal entities from the text into as complete timeline. Algorithm 1 is the main algorithm and uses supplementary algorithms 2, 3, 4 and 5.

Algorithm 1: Parsing Extracted Temporal Event Entities

Result: Extracted timeline of sleep events
Input: A tuple containing two lists: E list of event classes, and T list of normalized time expressions.

Initialize empty list TIMELINE.
if BED_TIME *not in* E *or* BED_TIME *in* E *is durative* **then**
    `bed_time` ← ask_bed_time()
end
else
    `bed_time` ← BED_TIME timex in T
end
Add `bed_time` to TIMELINE
if LIGHTS_OFF *in* E **then**
    if LIGHTS_OFF *is durative* **then**
        `lights_off` ← Add/Subtract LIGHTS_OFF timex in T to last time in recorded in TIMELINE
    end
    else
        `lights_off` ← LIGHTS_OFF timex in T
    end
    Add `lights_off` to TIMELINE
end

```
if SLEEP_TIME in E then
    if last time observed in Timeline is LIGHTS_OFF, SLEEP_LATENCY or BED_TIME then
        if SLEEP_TIME is durative then
            sleep_time ← Add/Subtract SLEEP_TIME timex in to last time in recorded in
              TIMELINE
        end
        else
            sleep_time ← SLEEP_TIME timex in T
        end
    end
end
else if SLEEP_LATENCY in E then
    sleep_time ← Add/Subtract SLEEP_LATENCY timex in T to last time in recorded in
      TIMELINE
end
else
    sleep_time ← ask_sleep_time()
end
Add sleep_time to TIMELINE
if WAKE_UP in E then
    if WAKE_UP is durative then
        WAKE_UP ← Add WAKE_UP timex in T to last time in recorded in TIMELINE
    end
    else
        wake_up ← WAKE_UP timex in T
    end
end
else
    wake_up ← ask_wake_up()
end
TIMELINE ← get_sleep_disturbed_patterns(TIMELINE, E, T, wake_up)
Add wake_up to TIMELINE
if OUT_OF_BED in E then
    if OUT_OF_BED is durative then
        out_of_bed ← Add OUT_OF_BED timex in T to last time in recorded in TIMELINE
    end
    else
        out_of_bed ← OUT_OF_BED timex in T
    end
    Add out_of_bed to TIMELINE
end
```

Algorithm 2: get_sleep_disturbed_patterns(TIMELINE, E, T, `wake_up`): Parsing sleep disturbance events

Result: TIMELINE
Input: Intermediate TIMELINE constructed E list of event classes, and T list of normalized time expressions.

Initialize $n_{range} \leftarrow$ wake_up − last time in TIMELINE
Initialize $n_{duration} \leftarrow$ `+00:05`
if *duration_of_disturbance in E* **then**
  if *range type* **then**
    $n_{range} \leftarrow$ DURATION_OF_DISTURBANCE timex in T
  end
  if *is type(e) == quantitative* **then**
    $n \leftarrow$ DURATION_OF_DISTURBANCE timex in T Add `sleep_disturbance` n times within n_{range} to TIMELINE
  end
  if *is type(e) == frequency* **then**
    $f \leftarrow$ DURATION_OF_DISTURBANCE timex in T Add `sleep_disturbance` at every f time within n_{range} to TIMELINE
  end
  if *durative type* **then**
    $n_{duration} \leftarrow$ DURATION_OF_DISTURBANCE timex in T
  end
end
for *e, t in (E, T)* **do**
  if *e* == SLEEP_DISTURBED **then**
    Add `sleep_disturbed` at t to TIMELINE
    if *next e == sleep_time* **then**
      Add `sleep_time` at next t to TIMELINE
    end
    else
      Add `sleep_time` at $n_{duration}$ to TIMELINE
    end
  end
end

Algorithm 3: ask_bed_time()

Result: `bed_time`
Input: Given:
Trained Transformer model $g()$ for translating the timex into a normalized form `bed_time` ← Get the bed time from the user via chatbot
`bed_time` ← g(`bed_time`)

Algorithm 4: ask_sleep_time()

Result: `sleep_time`
Input: Given:
Trained Transformer model $g()$ for translating the timex into a normalized form `sleep_time` ← Get the time of sleeping from the user via chatbot
`sleep_time` ← g(`sleep_time`)

Algorithm 5: ask_wake_up()

Result: `wake_up`
Input: Given:
1 Trained Transformer model $g()$ for translating the timex into a normalized form `wake_up` ← Get the time of waking up from the user via chatbot
2 `wake_up` ← g(`wake_up`)

Appendix D

Publications

[1] Fotedar. S, Vannisselroij. K, Khalil. S and Ploeg. B, *StoryTelling AI: A Generative Approach to Story Narration*, "IJCAI 2020 workshop: AI4Narratives"