Half Title

Title Page

LOC Page

To my dog
and my cat.

Contents

Foreword

I am delighted to introduce the first book on Multimedia Data Mining. When I came to know about this book project undertaken by two of the most active young researchers in the field, I was pleased that this book is coming in early stage of a field that will need it more than most fields do. In most emerging research fields, a book can play a significant role in bringing some maturity to the field. Research fields advance through research papers. In research papers, however, only a limited perspective could be provided about the field, its application potential, and the techniques required and already developed in the field. A book gives such a chance. I liked the idea that there will be a book that will try to unify the field by bringing in disparate topics already available in several papers that are not easy to find and understand. I was supportive of this book project even before I had seen any material on it. The project was a brilliant and a bold idea by two active researchers. Now that I have it on my screen, it appears to be even a better idea.

Multimedia started gaining recognition in 1990s as a field. Processing, storage, communication, and capture and display technologies had advanced enough that researchers and technologists started building approaches to combine information in multiple types of signals such as audio, images, video, and text. Multimedia computing and communication techniques recognize correlated information in multiple sources as well as insufficiency of information in any individual source. By properly selecting sources to provide complementary information, such systems aspire, much like human perception system, to create a holistic picture of a situation using only partial information from separate sources.

Data mining is a direct outgrowth of progress in data storage and processing speeds. When it became possible to store large volume of data and run different statistical computations to explore all possible and even unlikely correlations among data, the field of data mining was born. Data mining allowed people to hypothesize relationships among data entities and explore support for those. This field has been put to applications in many diverse domains and keeps getting more applications. In fact many new fields are direct outgrowth of data mining and it is likely to become a powerful computational tool.

Preface

Approximately 17 million people in the USA (6% of the population) and 140 million people worldwide (this number is expected to rise to almost 300 million by the year 2025) suffer from *diabetes mellitus*. Currently, there a few dozens of commercialised devices for detecting blood glucose levels [1]. However, most of them are invasive. The development of a noninvasive method would considerably improve the quality of life for diabetic patients, facilitate their compliance for glucose monitoring, and reduce complications and mortality associated with this disease. Noninvasive and continuous monitoring of glucose concentration in blood and tissues is one of the most challenging and exciting applications of optics in medicine. The major difficulty in development and clinical application of optical noninvasive blood glucose sensors is associated with very low signal produced by glucose molecules. This results in low sensitivity and specificity of glucose monitoring by optical methods and needs a lot of efforts to overcome this difficulty.

A wide range of optical technologies have been designed in attempts to develop robust noninvasive methods for glucose sensing. The methods include infrared absorption, near-infrared scattering, Raman, fluorescent, and thermal gradient spectroscopies, as well as polarimetric, polarization heterodyning, photonic crystal, optoacoustic, optothermal, and optical coherence tomography (OCT) techniques [1-31].

For example, the polarimetric quantification of glucose is based on the phenomenon of optical rotatory dispersion, whereby a chiral molecule in an aqueous solution rotates the plane of linearly polarized light passing through the solution. The angle of rotation depends linearly on the concentration of the chiral species, the pathlength through the sample, and the molecule specific rotation. However, polarization sensitive optical technique makes it difficult to measure *in vivo* glucose concentration in blood through the skin because of the strong light scattering which causes light depolarization. For this reason, the anterior chamber of the eye has been suggested as a sight well suited for polarimetric measurements, since scattering in the eye is generally very low compared to that in other tissues, and a high correlation exists between the glucose in the blood and in the aqueous humor. The high accuracy of anterior eye chamber measurements is also due to the low concentration of optically active aqueous proteins within the aqueous humor.

On the other hand, the concept of noninvasive blood glucose sensing using the scattering properties of blood and tissues as an alternative to spectral absorption and polarization methods for monitoring of physiological glucose

concentrations in diabetic patients has been under intensive discussion for the last decade. Many of the considered effects, such as changing of the size, refractive index, packing, and aggregation of RBC under glucose variation, are important for glucose monitoring in diabetic patients. Indeed, at physiological concentrations of glucose, ranging from 40 to 400 mg/dl, the role of some of the effects may be modified, and some other effects, such as glucose penetration inside the RBC and the followed hemoglobin glycation, may be important [30-32].

Noninvasive determination of glucose was attempted using light scattering of skin tissue components measured by a spatially-resolved diffuse reflectance or NIR frequency-domain reflectance techniques. Both approaches are based on change in glucose concentration, which affects the refractive index mismatch between the interstitial fluid and tissue fibers, and hence reduces scattering coefficient. A glucose clamp experiment showed that reduced scattering coefficient measured in the visible range qualitatively tracked changes in blood glucose concentration for the volunteer with diabetes studied.

Contributors

Michael Aftosmis
NASA Ames Research Center
Moffett Field, California

Pratul K. Agarwal
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Sadaf R. Alam
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Gabrielle Allen
Louisiana State University
Baton Rouge, Louisiana

Martin Sandve Alnæs
Simula Research Laboratory and
University of Oslo, Norway
Norway

Steven F. Ashby
Lawrence Livermore National
Laboratory
Livermore, California

David A. Bader
Georgia Institute of Technology
Atlanta, Georgia

Benjamin Bergen
Los Alamos National Laboratory
Los Alamos, New Mexico

Jonathan W. Berry
Sandia National Laboratories
Albuquerque, New Mexico

Martin Berzins
University of Utah
Salt Lake City, Utah

Abhinav Bhatele
University of Illinois
Urbana-Champaign, Illinois

Christian Bischof
RWTH Aachen University
Germany

Rupak Biswas
NASA Ames Research Center
Moffett Field, California

Eric Bohm
University of Illinois
Urbana-Champaign, Illinois

James Bordner
University of California, San Diego
San Diego, California

George Bosilca
University of Tennessee
Knoxville, Tennessee

Greg L. Bryan
Columbia University
New York, New York

Marian Bubak
AGH University of Science and
Technology
Kraków, Poland

Andrew Canning
Lawrence Berkeley National Laboratory
Berkeley, California

Jonathan Carter
Lawrence Berkeley National Laboratory
Berkeley, California

Zizhong Chen
Jacksonville State University
Jacksonville, Alabama

Joseph R. Crobak
Rutgers, The State University of New Jersey
Piscataway, New Jersey

Roxana E. Diaconescu
Yahoo! Inc.
Burbank, California

Peter Diener
Louisiana State University
Baton Rouge, Louisiana

Jack J. Dongarra
University of Tennessee, Knoxville, Oak Ridge National Laboratory, and
University of Manchester

John B. Drake
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Kelvin K. Droegemeier
University of Oklahoma
Norman, Oklahoma

Stéphane Ethier
Princeton University
Princeton, New Jersey

Christoph Freundl
Friedrich–Alexander–Universität
Erlangen, Germany

Karl Fürlinger
University of Tennessee
Knoxville, Tennessee

Al Geist
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Michael Gerndt
Technische Universität München
Munich, Germany

Tom Goodale
Louisiana State University
Baton Rouge, Louisiana

Tobias Gradl
Friedrich–Alexander–Universität
Erlangen, Germany

William D. Gropp
Argonne National Laboratory
Argonne, Illinois

Robert Harkness
University of California, San Diego
San Diego, California

Albert Hartono
Ohio State University
Columbus, Ohio

Thomas C. Henderson
University of Utah
Salt Lake City, Utah

Bruce A. Hendrickson
Sandia National Laboratories
Albuquerque, New Mexico

Alfons G. Hoekstra
University of Amsterdam
Amsterdam, The Netherlands

Philip W. Jones
Los Alamos National Laboratory
Los Alamos, New Mexico

Laxmikant Kalé
University of Illinois
Urbana-Champaign, Illinois

Shoaib Kamil
Lawrence Berkeley
Berkeley, California

Cetin Kiris
NASA Ames Research Center
Moffett Field, California

Uwe Küster
University of Stuttgart
Stuttgart, Germany

Julien Langou
University of Colorado
Denver, Colorado

Hans Petter Langtangen
Simula Research Laboratory and
University of Oslo, Norway

Michael Lijewski
Lawrence Berkeley National
Laboratory
Berkeley, California

Anders Logg
Simula Research Laboratory and
University of Oslo, Norway

Justin Luitjens
University of Utah
Salt Lake City, Utah

Kamesh Madduri
Georgia Institute of Technology
Atlanta, Georgia

Kent-Andre Mardal
Simula Research Laboratory and
University of Oslo, Norway

Satoshi Matsuoka
Tokyo Institute of Technology
Tokyo, Japan

John M. May
Lawrence Livermore National
Laboratory
Livermore, California

Celso L. Mendes
University of Illinois
Urbana-Champaign, Illinois

Dieter an Mey
RWTH Aachen University
Germany

Tetsu Narumi
Keio University
Japan

Michael L. Norman
University of California, San Diego
San Diego, California

Boyana Norris
Argonne National Laboratory
Argonne, Illinois

Yousuke Ohno
Institute of Physical and Chemical
Research (RIKEN)
Kanagawa, Japan

Leonid Oliker
Lawrence Berkeley National
Laboratory
Berkeley, California

Brian O'Shea
Los Alamos of The National
Laboratory
Los Alamos, New Mexico

Christian D. Ott
University of Arizona
Tucson, Arizona

James C. Phillips
University of Illinois
Urbana-Champaign, Illinois

Simon Portegies Zwart
University of Amsterdam,
Amsterdam, The Netherlands

Thomas Radke
Albert-Einstein-Institut
Golm, Germany

Michael Resch
University of Stuttgart
Stuttgart, Germany

Daniel Reynolds
University of California, San Diego
San Diego, California

Ulrich Rüde
Friedrich–Alexander–Universität
Erlangen, Germany

Samuel Sarholz
RWTH Aachen University
Germany

Erik Schnetter
Louisiana State University
Baton Rouge, Louisiana

Klaus Schulten
University of Illinois
Urbana-Champaign, Illinois

Edward Seidel
Louisiana State University
Baton Rouge, Louisiana

John Shalf
Lawrence Berkeley National
Laboratory
Berkeley, California

Bo-Wen Shen
NASA Goddard Space Flight Center
Greenbelt, Maryland

Ola Skavhaug
Simula Research Laboratory and
University of Oslo, Norway

Peter M.A. Sloot
University of Amsterdam
Amsterdam, The Netherlands

Erich Strohmaier
Lawrence Berkeley National
Laboratory
Berkeley, California

Makoto Taiji
Institute of Physical and Chemical
Research (RIKEN)
Kanagawa, Japan

Christian Terboven
RWTH Aachen University,
Germany

Mariana Vertenstein
National Center for Atmospheric
Research
Boulder, Colorado

Rick Wagner
University of California, San Diego
San Diego, California

Daniel Weber
University of Oklahoma
Norman, Oklahoma

James B. White, III
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Terry Wilmarth
University of Illinois
Urbana-Champaign, Illinois

Symbols

Symbol Description

α	To solve the generator maintenance scheduling, in the past, several mathematical techniques have been applied.
σ^2	These include integer programming, integer linear programming, dynamic programming, branch and bound etc.
$\sum$	Several heuristic search algorithms have also been developed. In recent years expert systems,
abc	fuzzy approaches, simulated annealing and genetic algorithms have also been tested.
$\theta\sqrt{abc}$	This paper presents a survey of the literature
ζ	over the past fifteen years in the generator
∂	maintenance scheduling. The objective is to
sdf	present a clear picture of the available recent literature
ewq	of the problem, the constraints and the other aspects of
bvcn	the generator maintenance schedule.

Part I

This is What a Part Would Look Like

1

Natural Language Processing

Author Name

CONTENTS

In this chapter we introduce modern NLP libraries, techniques and their applications. This chapter will focus on deep learning methods and less on computational linguistics that require nuanced knowledge of linguistics. We explore what it means to represent words and sequences of words with rich numeric representations that are better-suited toward modern computational tasks. We aim to capture some of these modern fine-tuned representations that are specially catered toward a semantic lexicon for medical language. We use these representations and aforementioned tools to showcase a modern reference implementation leveraging PyTorch, PyTorch Lightning and the Huggingface Transformers library. To wrap it all together, we walk through a complete ex-

ample that highlights best practices that encourage reproducibility and allow for systematic iterative improvements.

This includes:

- An introduction into fundamental NLP concepts
- Bootstrapping techniques to iterate on a dataset in the low-resource setting
- Storing of a reference dataset in a publicly-accessible location
- Downloading, caching, loading, splitting, and preprocessing of the data
- Setting up of a cloud-based GPU workstation (?) (–this might be overkill for now, but keep if we can)
- VSCode (?)
- Monitoring the training run:

  Logging and experiment tracking

  Learning curves

  Metrics
- Hyperparameter tuning, some tricks of the trade
- Offline evaluation and sanity checking

We will keep the discussion focused on tasks in epilepsy, using a running example of SUDEP prediction from electronic medical record (EMR) notes. Many of the concepts introduced here are very general and are straightforward translations to domains outside of SUDEP prediction, epilepsy, and even NLP.

1.1 Introduction to Natural Language Processing

Natural language processing (NLP) is a field of computer science that deals with the extraction, processing, and understanding of human language. It is known as the field of computer linguistics, and is a subfield of artificial intelligence. Common NLP tasks include sentence segmentation, tokenization, part-of-speech tagging, named-entity recognition, parsing, question answering, summarization and classification.

How can we teach a computer to perform these tasks? The first challenge is that computers, at their core, only understand numbers. So first we need to think about how words can be represented with numbers such that we can perform calculations on them. The numbers should allow the computer to assign meaning to words and their context with other words around it.

There is no perfect way to represent language in this numeric fashion. The best we can do is to capture representations that we feel capture the *syntactic* features of text, as well as *semantic* features of text. Syntactic information refers to grammatical constructs and the way that words are put together in a language. Semantic information refers to the meaning of this body of text. Such distinctions are fundamental concepts in computer science and date back to the very foundations of information theory. [**?**]. Much of this chapter, as well as much of the current work in the field of computational linguistics is centered around finding better and better representations that capture both types of this information.

How do we begin to choose these representations? Do we choose to represent words, or perhaps individual letters? Do we choose all the words? What do we do with punctuation and numbers?

1.1.1 Vocabularies and Words

In NLP, the set of unique words in a corpus is called a *vocabulary*. While *the* is the most common word in the English language, it is only one entry in a very large vocabulary table. A rare word like *hemispherectomy* is also one entry in this table. Both *apple* and *apples* might be separate entries in our vocabulary table. Generally the first step in many NLP systems is defining this vocabulary and the rules behind which words are in this table and which words are not.

The process of splitting text into separate smaller units, in this case words, is known as *tokenization*. When we run many NLP tasks, we almost always preprocess our text by putting it into one of these tokenizers. The output tokens, in this case words, are the indexes in our lookup table to different numeric representations that can be understood by a machine. There are a number of great tokenization libraries that are open source, and you should always use them rather than try to implement this yourself.

```
# Example 1: Spacy Tokenization Using Medical Words

import spacy
sample = "After a temporal lobe resection, the atonic and clonic seizure frequency fell

nlp = spacy.load("en_core_web_sm")
english_vocab = set(nlp.vocab.strings)
spacy_document = nlp(sample)

for token in spacy_document:
    found = "found" if token.text in english_vocab else "NOT FOUND"
    print("{}: {}".format(token.text, found))
```

The discrete numeric entities created in Example 1 are the indexes for our

representations. The tokenizer system did much of the magic under the hood, lowercasing and standardizing text and creating special tokens for punctuation and numbers.

When we build a vocabulary V, we define a preset vocabulary size $|V|$. There is a tradeoff in your choice of $|V|$. If you choose a number that is too small, then your system will only recognize a few words and lose a lot of important task-specific vocaublary. However, if you include the entire english language, your system will be slow, expensive, and not perform as well. Commonly this number is set around 20,000 - 40,000 words in English, though it can be much larger.

Often a vocabulary will include special tokens that make our lives easier when working with NLP tasks. Common special tokens include:

EOS - An end of sentence/input marker.

PAD - Special inputs to ignore, usually following an EOS marker.

SEP - A separator, which can be used in inputs that contain multiple sentences or samples.

OOV - Out of vocabulary, also commonly represented as [UNK] (unknown), which is used when we come across a word that does not exist in our vocabulary.

The out-of-vocabulary token is of particular importance in medicine. If we load a vocabulary with 30,000 words, often these are roughly the most common 30,000 words in the English language. We will still have a few [OOV] tokens for rare words. However, just because a general system chooses its vocabulary based on word frequency does not mean that this is the best choice of vocabulary for the task at hand. Medical corpora have a very different word frequency distribution than non-medical corpora, particulary because of nuanced vocabulary that is absolutely crucial.

In Example 1 above, notice the importance of the words that have been thrown away. If this sentence were to be loaded into a machine as-is, we might be losing far too much information because we have to nearly all of the relevant terminology with OOV tokens.

We will address this issue in the ensuing sections. For now, suffice it to say that we should make sure our vocabularies are catered toward our task at hand or are built in such a way that they can still extract signal from these tokens.

1.1.2 Word Representations

Given that we have defined a vocabulary V of words, these are the keys to our lookup table of numeric representations. But what do we store in that table as the value?

A simple way to achieve this representation for V is to assign each word

a unique integer i. The word is then represented as a vector w of length $|V|$ with all zeros and a one at index i[1].

```
have = [1, 0, 0, 0, 0, 0, ... 0]
a    = [0, 1, 0, 0, 0, 0, ... 0]
good = [0, 0, 1, 0, 0, 0, ... 0]
day  = [0, 0, 0, 1, 0, 0, ... 0]
...
```

We could now represent a sentence as the sum of the word vectors $S = \sum_j w_j$. This representation is suitable as input for any classification algorithm (such as logistic regression or a decision tree), to make a prediction about our target variable, e.g. whether the sentence is relevant to our epilepsy task. This simple representation fulfills our requirements but comes with some drawbacks:

- We implicitly assume that each word in the sentence is equally important.
- Each word is equally similar to every other word (e.g. by taking the euclidian distance between word vectors).
- The representation is invariant to reordering of the words.

The first point is a problem, because as we add more word vectors together, the sentence reperesentation will converge to the global average and drown out any signal relevant to the specific sentence at hand. To address this we can instead write a weighted sum $S = \sum_j \lambda_j w_j$, where each word vector is weighted by its importance λ_j, and there are statistical methods we can use to compute an importance value[2]. For example a word like *the* is very common and appears in many different contexts. It is unlikely that there is a lot of signal we can extract from it. On the other hand, a word like *epilepsy* will be much more rare and specific to a given task, and we would like to raise its importance. This has the net effect of allowing us to find a good representation for larger word sequences.

To address the second point, we will be touching on a very important concept, not just relevant in NLP, but also in the world of ML in general: Embeddings.

1.1.3 Embeddings

An embedding is a representation of a vector space that captures the similarity between the entities we are encoding, where entities can be words, images, e-commerce products, movies, houses, and many other data modalities. Scientific progress in the field of ML is often directly about finding algorithms

[1] This is also known as a one-hot encoding
[2] TF-IDF

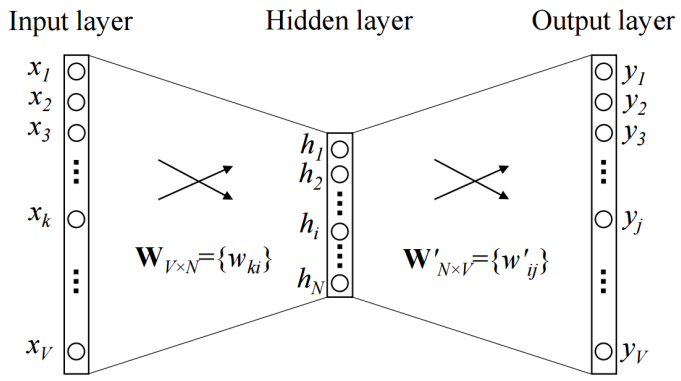

Figure 1: A simple CBOW model with only one word in the context

FIGURE 1.1
Word2vec

that can learn high-quality robust embeddings in an efficient and scalable way, and the final performance of a given architecture is largely determined by the quality of the embeddings it learns.

To illustrate this concept, we will be talking briefly about one of the earliest algorithms that has been used to learn embeddings for words: *word2vec* [**?**], a technique pioneered by a team led by Thomas Mikolov. The idea behind *word2vec* is to set a word into context with the words surrounding it. In NLP, the study of techniques that determine the probability of a given sequence of words is known as language modeling.

For example the word *epilepsy* may appear in the context of words like *symptom*, *medication*, or *episode* and vice versa. However, we would not expect that *epilepsy* would appear in the context of words like *volleyball*.

We then translate this idea into a training task: Given the sum of the one-hot encoded word vectors of the words surrounding the target word, map it to the one-hot encoded vector of the target word.

This is a task that can be solved by a neural network with the following setup: Add a layer l_1 that maps from the input dimension (of size $|V|$) to an intermediate dimension of size h and then another layer l_2 that maps from the intermediate dimension to the output dimension (of size $|V|$). Here h is what is called embedding dimension and $h << |V|$. While h is generally a hyperparameter that can be optimized at a later stage, a reasonable choice is to use a value according to the rule of thumb:

$$h \approx 4\sqrt[4]{|V|} \tag{1.1}$$

For a vocabulary size $|V| = 30000$ this comes to a value of about 50. During

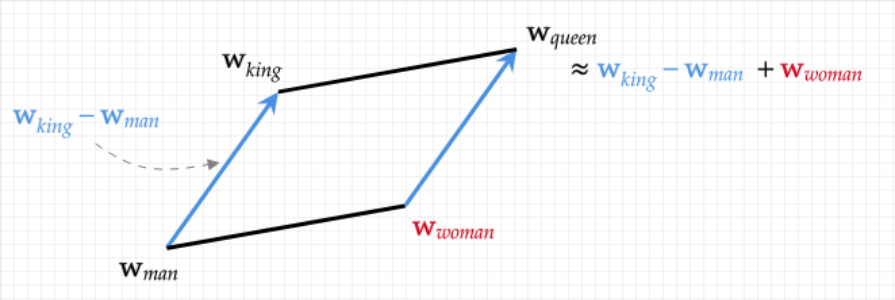

FIGURE 1.2
Word2vec

training, the network will learn to encode the original sparse indicator vectors in a way that preserves the maximum amount of information necessary to predict the target word, while being forced to squeeze the information through the low dimensional bottleneck. An interesting side effect of this approach is that we can use part of the network to encode a given word in V by using for example the inverse of l_2 as a lookup table. The vectors that we obtain from this lookup table are called word embeddings and they have some very useful properties:

- Similar words tend to be close together (e.g. in terms of euclidian distance).
- Averaging word embeddings of a sentence will give us a more robust representation than our naive approach.
- We can perform some arithmetic, such as adding and subtracting the embedding vectors to traverse the space.

These properties also imply that the represention for two sentences will be close together if they are semantically similar, which will make the job of, say, a downstream classifier much more straightforward - it only has to slice the embedding space. The hard part of understanding the meaning of the sentence is already done. Since the training data can be generated from just raw text, it is sufficient to train the embeddings once on a very large corpus[3] and then reuse them, in effect giving us a headstart when building task-specific models. Instead of also having to learn what words mean, and some of the syntactic rules behind written language, we can get a head start on this and focus our computational resources on adapting this knowledge to a specific task.

This underlying assumption that similar neighboring words share a semantic similarity to center words is powerful, and at the heart of many modern training objectives for embeddings that are used in many downstream ML systems. We do not inherently inject a great deal of model bias, but rather

[3] GloVe

rely on having a large enough training set that the word vectors will become meaningful representations with respect to one another. A corpus such as Common Crawl[4], which contains roughly 840 billion tokens of over two million unique words, provide such scale. There are other corpora used in these unsupervised processes, one such common candidate being a collection of all Wikipedia articles.

In practice, one training objective can be to use the distributed representation of a set of context words to guess a center word. This is known as the continuous bag of words (CBOW) model.

Consider a training document describing patients with epilepsy, where we are attempting to train a model with a predefined vocabulary V of the top For a vocabulary V, where we have chosen the top $|V|$ most common words in English to learn. The document being chosen in this training iteration may contain the following excerpt:

```
... recommended administering Lamictal to treat their
focal seizures, the patient continued experiencing
symptoms though reported a 50% reduction in ...
```

In training our word embeddings, we will randomly mask one of the words, in this case *patient*. Our model makes use of an additional parameter, a context window size c, that helps to provide necessary context. If we choose $c = 8$, our training sample would become

```
... lamictal to treat their focal seizures, the [MASK]
continued experiencing symptoms though reported a 50 % ...
```

For a given center word at index i, we consider previous words at indexes $i-1$, $i-2$, ..., $i-c$, and all subsequent words at indexes $i+1$, $i+2$, ..., $i+c$. In our word sequence S, we may then choose to sum the word vectors of each of these context words. Our context input can be written as:

$$\sum_{\substack{j=-c \\ j\neq 0}}^{c} w_{i+j} \tag{1.2}$$

Our task is then to predict a probability distribution over the size of our vocabulary $|V|$, where the target is a vector of all 0's with a 1 at the index in the vocabulary for the word *patient*.

There are a few additional takeaways from this example. Note that we will address many of these in the sections to follow.

- The training process will likely consider this text many times, and mask out different words in each iteration.

[4]Common Crawl Link

- Our processed sentence is slightly modified from our original input. Notice that the number and percent symbol are split, and that Lamictal is lowercased.
- The word *patient* can also be an adjective. Our *patient* embedding will also capture this meaning in our model, dependent upon how often it is used with that meaning.
- If our context window is too small, we lose information. If it is too large, our signal becomes fuzzy as we converge to the global mean vector.

What if we choose to use only a center word, and have our model predict all of the surrounding context words? This is a common training technique for embeddings as well, and is known as the Skip-Gram model.

1.1.3.1 Spacy Example

This is placeholder text

1.1.3.2 Character and Subword Embeddings

In Example 1, we covered the issue of missing vocabulary that is frequently encountered when using word-level tokenization. However, there are other strategies that are commonly used that help to avoid this issue. One such simple way around out-of-vocabulary tokens is to use character-level tokenization. Instead of having a vocabulary that is tens of thousands of words long, you only have a vocabulary that contains the characters in your language, with the addition of punctuation, digits, and a few special tokens for sentence and word marking. As long as tokenizer is able to mark which characters start a word and which characters are continuations of a word, then the original sequence of words can always be recovered.

Unfortunately, the tradeoff for a small vocabulary size with no unknown word entries is that the length of tokens to represent a sentence is clearly much larger. Empirically, these models do not perform as well, partially because of this sequence but also because of the loss of information that is represented in word vectors.

In practice, the best models are often somewhere in the middle, in what is known as word-piece, or subword, vocabularies. These tokenizers and vocabularies contain a rich set of common words, but also have subword units that can be used to piece together out-of-vocabulary words. These subword units behave like normal word vectors and can store semantic and syntactic information.

```
# Example 2 - Wordpiece Tokenizers

from transformers import BertTokenizer
```

```
sample = "After a temporal lobe resection, the atonic and clonic seizure frequency fell

bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

encoded_ids= bert_tokenizer.encode(sample)
encoded_tokens = bert_tokenizer.convert_ids_to_tokens(encoded_ids)

for token, id  in zip(encoded_tokens, encoded_ids):
  print("{}: {}".format(token,id))
```

In example two,

1.1.4 Transformers

There have been significant advancements in ML due to a variety of factors, such as the availability of larger and larger datasets, as well as ever increasing computational power at lower and lower prices, better software tools, libraries and model architectures. Especially the advent of deep learning (DL), fueled by the emergence of GPUs from about 2012, enabled scaling to many orders of magnitude larger datasets and models. The common wisdom before DL was that as the number of parameters in a model increases beyond a certain threshold, it tends to overfit the training data, leading to poor generalization when deployed in the real world. Deep Learning based models empirically do not seem to exhibit this behavior, and instead tend gain performance as they get bigger, albeit more slowly.

Until recently, building strong NLP systems required deep understanding of language and its structure, as well as large amounts of data, even when relying on pre-trained word embeddings - the resulting systems were often still brittle.

In recent years some surprisingly powerful architectures and pre-training regimes have emerged that build on the concepts we have discussed here, most notably under the name of *Transformers*[5], that address the last bullet point of our representation's shortcomings.

The details of the inner workings are beyond the scope of this tutorial, for now it suffices to know that they are able to learn robust sentence embeddings with a fine grained understanding of syntactic structure, semantics, and general knowledge of the world. In fact, they can be considered near the level of a human that just knows the English language (or other languages), along with a broad array of factual knowledge about the world (think Wikipedia). Just like a human, they can be taught specialty knowledge that is relevant to a given task, in ML we would say we *finetune* them. This paradigm shift has led to a step function improvement in terms of performance and sample efficiency in a wide variety of NLP tasks where Transformers are the defacto state of the art.

[5] BERT, GPT

In the following sections, we will discuss a broad set of problem statements Transformers are suitable for, along with a set of techniques a practitioner can employ to arrive at robust solutions, even with limited data.

1.1.4.1 Pre-training

Modern DL based systems, such as the aforementioned Transformer architectures, are pre-trained on gigantic datasets[6][**?**] and can be downloaded for free[7]. Starting with a model pre-trained on a broad set of topics significantly reduces the amount of task specific training data required to achieve a given performance. Furthermore, it may be helpful to start with a model that is pre-trained on more domain specific datasets such as `BioBERT[1]` or `Bio_ClinicalBERT[?]`, or pre-train your model from scratch.

1.2 NLP Tasks in Epilepsy

Now that we have an understanding of some of the basics of computational representations of language data, let us take a look at some of the modern use cases for NLP in industry, catered toward epileptologists and other medical practitioners.

1.2.1 Unstructured Text - Retrospective Research

Within the medical field, there are many sources of unstructured text that are primarily aimed as a either a communication channel between specialists, or a way for patients to pass information that isn't captured in a structured format. Clinical Notes, Progress Notes, Operative Notes, Discharge Summaries, Pathology Reports, Surgery Reports; all of these sources carry information, but traditionally require experts in the field to extract it. If we can teach systems to read through this unstructured text and perform respectably close to these experts, we can perform large-scale clinical retrospective research much faster and at a much lower cost.

Example: https://academic.oup.com/jamia/advance-article/doi/10.1093/jamia/ocac018/6534112

1.2.2 Unstructured Text - Quality of Life

This is a placeholder for some of the work that is done on the Quality of Life surveys for the INJS Seizure Tracker datasets.

[6] BooksCorpus (800M words, Wikipedia 2,500M words)
[7] Huggingface

1.2.3 Text Preprocessing and Tokenization

Placeholder.

1.2.4 Named Entity Recognition and PII

This is a working title. NER is an important problem in medicine, for PII if nothing else, and I figure we could go through some basics of using it with Spacy, NLTK, or Stanford NLP here.

The reluctance towards data sharing and common formats in the medical domain has hampered progress significantly.

Stopwords would be worth mentioning?

1.2.5 Mining Unstructured Text and Clinical Documentation

This is placeholder text. Talk about clinical documentation and EHR notes, and other forms of unstructred text, being used for downstream classification tasks. Then, give some examples of them, and what these people did, and possibly how transformers could probably improve this work greatly.

1.2.6 Clinical Trial Matching and Eligibility Prescreening

This is a placeholder text. https://www.nature.com/articles/d41586-019-02871-3

1.2.7 Surgical Candidacy

This is placeholder text

1.2.8 Avoiding Epilepsy Misdiagnosis

This is placeholder text. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4419916/

1.3 Full Walkthrough - SUDEP Detection

This is placeholder text

1.4 Full Walkthrough - Financial Impact of Epilepsy

This is placeholder text

Bibliography

[1] 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. *CoRR*, abs/1901.08746, 2019.