Masterthesis

Knowledgebase construction of genetic variants in literature

Philip Schledermann

12.09.2018

Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik

Bearbeitungszeitraum
12. 03. 2019 – 12. 09. 2018

Gutachtende
Prof. Dr. Hannah Bast
Prof. Dr. Rolf Backofen

Betreuer
Dr. Thomas Schödl

You shall know a word by the company it keeps - Firth, J. R. 1957:11

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Inhaltsverzeichnis

Zusammenfassung **1**

Abstract **3**

1 Introduction **5**
1.1 Motivation . . . 5
1.1.1 Mutations and their implications . . . 6
1.1.2 Identifying mutations in text . . . 8
1.2 Terminology . . . 9
1.3 Guiding example . . . 9
1.4 Goals . . . 10
1.5 Related work . . . 11

2 Fundamentals **13**
2.1 Named Entity Recognition (NER) . . . 13
2.2 Named Entity Normalization (NEN) . . . 13
2.3 Text cleaning and simplification . . . 14
2.4 Vector Space Model . . . 14
2.4.1 Vector encoding of words . . . 14
2.4.2 word2vec . . . 15
2.5 Knowledgebase construction . . . 17
2.6 Full text . . . 17

3 Model creation **21**
3.1 Preparation of the data . . . 21
3.1.1 Uploading the raw data to the cluster . . . 21
3.1.2 Extracting the data from the XML files . . . 22
3.1.3 Deduplication of the data . . . 22
3.1.4 Tokenization of the text . . . 24
3.1.5 Apply TERMite and Stanford CoreNLP . . . 24
3.2 Model creation . . . 25
3.3 Knowledgebase creation . . . 25

4 Information Extraction **29**
4.1 Named Entity Recognition . . . 29
4.2 Word Embeddings . . . 29
4.3 Similarity . . . 31

5 Implementation **33**
5.1 Tools . . . 33
5.1.1 python and libraries (ETree) . . . 33
5.1.2 Apache Parquet, Hadoop and Spark . . . 33
5.1.3 SciBite TERMite . . . 35
5.1.4 Stanford CoreNLP . . . 35
5.1.5 N-Gram creation . . . 35
5.1.6 WordEmbeddings . . . 36
5.2 Data . . . 36
5.2.1 PubMed . . . 36
5.2.2 PubMedCentral . . . 37
5.2.3 ScienceDirect by Elsevier . . . 37
5.3 Runtime and Hardware . . . 37
5.3.1 Deduplicating PubMed and PMC . . . 38
5.3.2 Model preprocessing . . . 38
5.3.3 Word Embedding construction . . . 39
5.4 Evaluation data . . . 40

6 Results **45**
6.1 Evaluation . . . 45
6.2 Error Analysis . . . 46

7 Discussion **51**

8 Outlook and future work **53**

Danksagung **55**

Bibliography **57**

Zusammenfassung

In Publikationen werden oftmals synonyme Begriffe für eine genetische Variation verwendet. Für die verbesserte Suche nach Veröffentlichungen über genetische Variationen in wissenschaftlichen Publikationsdatenbanken wie PubMed und PubMedCentral, sollen die unterschiedlich verwendeten Begriffe auf die präferierte Bezeichnung der jeweiligen genetischen Variation zurückgeführt werden. Mit ClinVar existiert eine manuell kurierte Datenbank, in der die Begriffe welche in Standardnomenklatur geschrieben werden enthalten sind. Um auch Begriffsvariationen zusammenzuführen, welche nicht der Norm entsprechen sollen diese aus den Veröffentlichungen extrahiert werden.

Der Text wird mittels Named Entity Recognition und anderen Algorithmen der Sprachverarbeitung vereinfacht.

In einem "word embedding"Modell werden aus dem Text Begriffe und ihre Kontexte extrahiert und miteinander verglichen. Gleiche Kontexte für unterschiedliche Begriffe weisen darauf hin, dass es sich bei den Begriffen um Synonyme handeln könnte. Dieses Modell wird auf unterschiedliche Schreibweisen von genetischen Variationen angewandt.

Das Modell wird gegen ClinVar evaluiert. Hierbei zeigt sich, dass der überwiegende Teil der normierten Standardbegriffe teils verschwindend gering in der verwendeten wissenschaftlichen Literatur enthalten ist. Zudem zeigt sich, dass das Modell nicht ohne zusätzliche Filterung der Ergebnisse verwendet werden kann.

Abstract

In publications often synonymous terms are used for genetic variations. Improving the search results for genetic variants in scientific publication databases like PubMed and PubMedCentral, the different terms used in publications shall be linked to their preferred name. ClinVar is a manually curated database in which terms in standard nomenclature are stored and linked together. To link mutation mentions that are not written obeying the nomenclature, the mutation mentions are extracted from the text.

As part of this thesis the text is simplified. This is done using Named Entity Recognition and other algorithms of natural language processing.

In a word embedding model the terms and their contexts are extracted and compared. Similar contexts but different terms indicate that the terms are synonyms. The model is applied to various spellings of genetic variants.

The model is evaluated against ClinVar. It is shown that most of the terms following the standard nomenclature are not present in the analyzed scientific literature. Additionally it is shown that the model is not applicable without further filtering of the results.

1 Introduction

1.1 Motivation

Personalized Healthcare is a new step in drug development as it sees the patient as an individual. Formerly patients have been separated into different groups, by looking at causes of a disease. With this more fine grained approach it is possible to use targeted medicines to treat diseases in a more effective way ([DS14], [BTZ+12]). This should result in fewer adverse events and better patient outcomes. For achieving this goal, researchers, pharmaceutical companies and clinics are intensely working on identifying the disease differences on a molecular level. Especially in cancer more than 250 of these molecular subtypes have been identified [Ltd18]. With the information about the drivers of a disease, doctors are enabled to select the treatment with the highest success probability for the patient.

The research in these topics is performed by many different scientists in different clinics, laboratories and companies all over the world. One common point is that the results are published in scientific papers. An extremely valuable resource for the research results can be found in PubMed, a public corpus provided by the National Institute of Health (NIH). It is explained in more derail in subsection 5.2.1. PubMed contains about 27 million abstracts and is growing daily [MU17]. Every day about 2400 entries are added to PubMed which is overloading the manual curation of literature [RE15]. Querying this corpus with an entity name is not sufficient, as for many entities in the biomedical domain more than one term exists. Although there are curated databases using standardized nomenclatures, they can not be used easily, because entities and especially mutations are often spelled differently.

With Information Extraction (IE) systems it is possible to build a knowledge base out of textual data. Therefore it is essential to extract these entities at first in a Named Entity Recognition (NER) step and to normalize these detected entities to common names in a Named Entity Normalization (NEN) step. Having normalized entities enables researchers to ask quantitative queries about successful experiments without the necessity to search for all available text or searching for all possible synonym combinations.

In the biomedical domain there are various entities of major importance. For the scope of this work the following three entities are of central importance:

- Gene
- Mutation

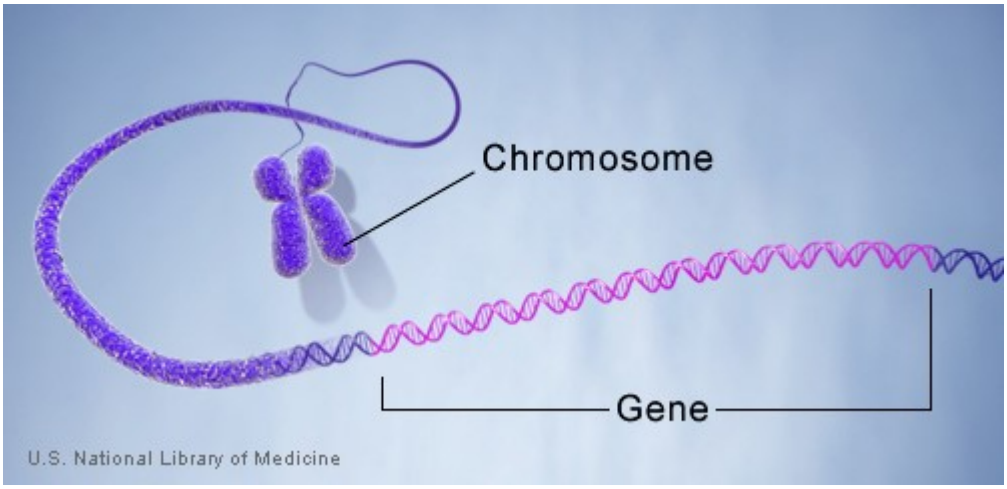
- Disease

Other entities of high importance like drug, protein or institute have not been used in the context of this work.

1.1.1 Mutations and their implications

In humans, each cell normally contains 23 pairs of chromosomes, for a total of 46. These chromosomes are made of DNA that has about 3 billion base pairs. A gene is called a certain sector of the DNA (see Figure 1.1). The DNA is built by a sequence of four different nucleotides. Each nucleotide contains of one of four nucleobases: adenine (abbreviated by A), cytosine (C), guanine (G) and thymine (T). The bases are always pairs of A with T, C with G that bind with each other on the opposite DNA strand.

Figure 1.1: Genes are sections of the DNA which is winded up in chromosomes. [MUGHR13b]



The information encoded in a gene is decoded in a two step process: transcription and translation [ABH+12]. In a first step information from the DNA is transcribed into an RNA sequence. In this process the information of the DNA is transferred into a different chemical format but still based on nucleotides. Figure 1.3 shows this in the upper part. The bases are the same only Uracil (U) is used instead of T. In the translation process the RNA is decoded into a protein. Each triplet of bases encodes for a specific amino acid. The lower part of Figure 1.3 is showing on the left side the translation of the triple 'G', 'A', 'G' to the amino acid glutamatic acid.

Each cell contains a copy of the same DNA. However in some cases there are appearing variations of this DNA e.g. through replication errors, external influences or cancerous cells. When these errors occur on a certain position of the DNA and only affect a single base pair the mutation is called a 'Single Nucleotide Polymorphism'

(SNP). SNPs are often written down as <reference amino acid><position offset from the start of the gene><mutated amino acid> (e.g. 'T1799A'). They are often prefixed with a 'c.' to indicate that the mutation is explicitly occurring on the c-DNA level, a special form of DNA. Through transcription and translation processes triplets of the DNA encode for one protein. The description of a mutated protein is similarly as the one of mutations on DNA level: <reference protein><position offset of proteins from the start of the gene><mutated protein> (e.g. 'Val600Glu'). In Figure 1.2 an illustration of a SNP is shown. In this case G is replaced by A.

Figure 1.2: A Single Nucleotide Polymorphism (SNP) - A single base pair is changed compared to the reference. In this example Guanine (G) with its base pair partner Cytosine (C) is replaced by Adenine (A) with its base pair partner Thymine (T) [MIS14]

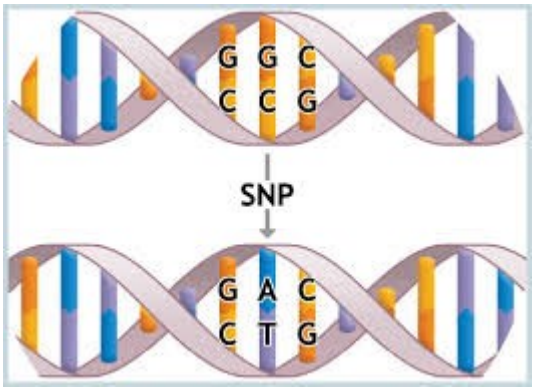


Figure 1.3 shows the mutation 'c.20A>T'. This can be read as A is exchanged by T at position 20 on the DNA level. With the transcription process the DNA is transcribed to RNA and later translated to proteins. On the protein level the DNA mutation is reflected into two different proteins: Glutamatic acid is constructed from the reference DNA and Valine is the amino acid constructed from the mutated form. This mutation happens at the offset 6 and is therefore written down as: 'p.Glu6Val'. There exists also single letter code for amino acids; in this case it would be: 'p.E6V'. The prefix 'p.' explicitly denotes that a mutation on protein level is mentioned.

The consequences of different proteins from Figure 1.3 are shown in Figure 1.4 where the abnormal form of the protein (hemoglobin) is sticking together and forms chains [MUGHR13a]. Because of this clumped hemoglobin the red blood cells are not in the same shape but like a sickle shape as shown in Figure 1.5. Because of this sickle shape the blood cells cannot float around like normal blood cells which can result in blockades [Pal18]. This disease is called sickle cell anemia.

In the previous example already three different types of writing down a mutation have been presented: 'c.20A>T', 'p.Glu6Val' and 'p.E6V'. All of these three mentions are referring to the same mutation and act as synonyms. In case of 'c' and 'p' the authors might want to focus on either DNA or the protein level but principally refer to the same mutation but on different abstraction levels.

Figure 1.3: In this case one Single Nucleotide Polymorphism (SNP) alters the proteins constructed from the DNA: here Adenine is replaced by Thymine in the DNA, through transcription and translation processes proteins are constructed from the information encoded in the DNA. Through the different DNA information different proteins are constructed: Valine instead of Glutamic acid is integrated at position 6 of the protein. This leads to abnormal protein form and function (in this case hemoglobin) [Pal18].

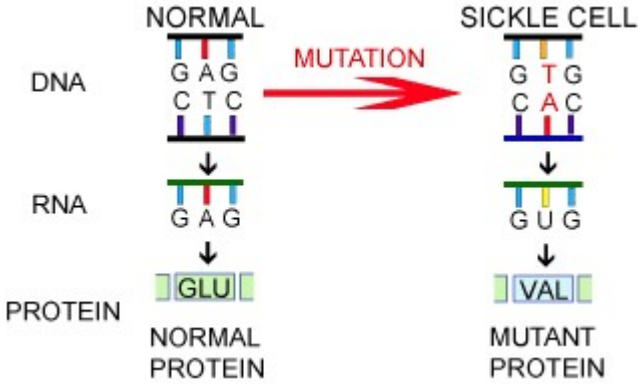
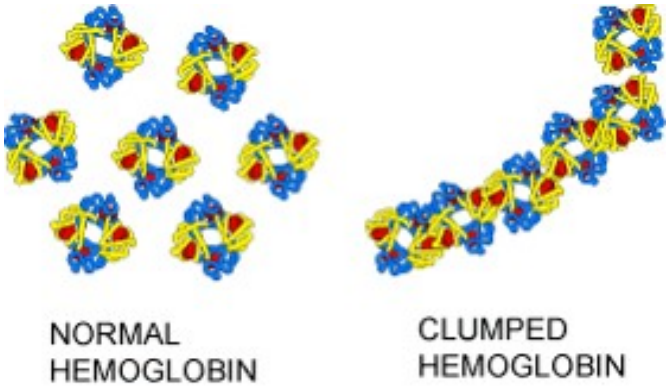


Figure 1.4: Abnormal hemoglobin sticking together. [Pal18]

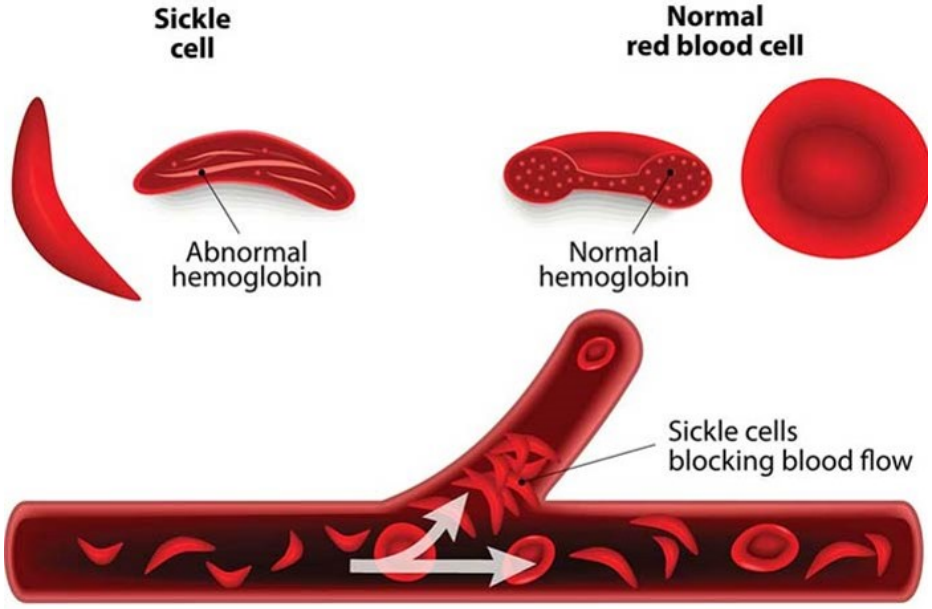


When querying a knowledge base containing genetic variants by 'c.20A>T' also documents containing 'p.E6V' shall be retrieved.

1.1.2 Identifying mutations in text

In the past there has been great success in recognizing and normalizing genes and diseases by using Machine Learning models, for example in [BDS+08] and [UBP17]. The results of using Machine Learning models for the extraction of mutations have been published this year and the year before in [CBU+], [WPF+17], [LKC+18]. These approaches are focusing on identifying a mutation. The normalization of the found mutation is then done by regular expressions and database lookups like in

Figure 1.5: Through the abnormal hemoglobin chains the sickle cells do not have the same structure which results in blocking of the blood flow. [She18]



[WPF+17] and [TRH+16].

In this thesis a system is shown to identify mutation mentions as well as normalizing them to a preferred label without using rule based extraction.

1.2 Terminology

For the process of linking different entities to a standard concept the terminus "Named Entity Normalization" (NEN) is used. NEN is also used by other publications in this area like [TRH+16], [TKF+11], [VLBW+13] and [WPF+17].

1.3 Guiding example

In this thesis an example is used to support the understanding of different concepts for readers not familiar with the topic. The example reappears in different sections. With this example and its explanations it is not intended to give a complete explanation of the biological concepts. It applies simplifications whenever the details are not necessary for this thesis.

The example is based on the BRAF gene. A study has shown that patients carrying the V600E mutation of BRAF react to a drug resulting in a partial or complete decrease of the metastatic melanoma tumor [DS14] [FPK+10].

'V600E' is a mutation on the protein level and the following other names are used as synonyms:

- p.V600E : The short form but referring to the protein level explicitly
- Val600Glu : The long form of the protein level mutation without the explicit mention of the level
- p.Val600Glu : The long form of the protein level mutation using three letter codes for the proteins and meaning the protein level explicitly
- c.1799T>A : The DNA level mutation at the position 1799 where T is replaced by A and meaning the cDNA level explicitly
- 1799T>A : The DNA level mutation without meaning the level explicitly
- T1799A : The DNA level mutation by using both; a common and ambiguous naming
- rs113488022 : A standardized identifier contained in the dbSNP database [SWK+01] (a SNP database containing mutation information for several species)

Exemplary a search in PubMed for V600E returns among others the article with the PubMed ID '30013664' [AMAS18], where only V600E is used to describe the mutation. While a query using '1799T>A' results in an article list containing the article with the PubMed ID '29808165' [FRHLL+18]. Each article is not appearing in the search results of the other.

With the eight terms acting as synonyms, querying for one term should return also documents containing the synonyms.

1.4 Goals

A knowledge base shall be created, in which researchers can search for genes, their mutations and diseases. The results shall be presented in a tabular format, where each row represents a hit of a gene, a mutation or a disease on a sentence level. The genes and diseases shall be tagged with standard tagging tools. For mutation mentions there shall be used a tagging and normalization method that allows a high recall. In addition to the knowledge base, a dictionary of genetic mutations and their different spellings should be created. This shall enable mutation tagging and normalization on other corpora. In the context of the described tasks the usage of word embeddings for creating synonym candidates shall be evaluated. As input three large corpora shall be used. First, PubMed, containing over 27 million scientific abstracts (it is explained in more detail in subsection 5.2.1). Second, PubMedCentral, the OpenAccess subset contains 1.7 million full text articles (see subsection 5.2.2). Third, ScienceDirect, a subcorpus containing about 2 million full text articles (see subsection 5.2.3). PubMed and PubMedCentral are publicly hosted by the National Institute of Health (NIH). ScienceDirect is licensed from the publisher Elsevier.

1.5 Related work

The extraction of mutations from public texts has been the topic of several research articles that were published recently. Retrieving mutation mentions in text is a topic that has been investigated for several years. As one of the still common baselines MutationFinder can be counted [CBR+07]. Therefore it is also referred to MutationFinder as a baseline. MutationFinder scans through the text and applies regular expressions to extract mutations from its input. It has been successful in combining multiple of these rule-based systems and showed an increase in performance compared to using a single system in recognizing mutations[YV14]. All of these systems share that they are rule-based extraction tools that do not normalize the found mutations.

In 2011 [TKF+11] there has been a proposal for a rule-based normalization procedure. Combining recognizing and normalizing mutation mentions, a recent system, SETH [TRH+16], was the first open-source system. The system tmVar2.0, proposed in [WPF+17], can be regarded as the first system that combined machine learning techniques using Conditional Random Fields (CRFs) to recognize a mentioned mutation and to apply normalization. This normalization process is similar to the method proposed in [TKF+11] and applied in [TRH+16] as it applies regular expressions on the found mutation mention and matches them against a database. In [CBU+] the focus is on natural language text mining with some mutations being written in verbose language (e.g. 'Valine is replaced by Glutamatic Acid at position 600'). The method proposed uses word embeddings to improve the performance. The system for mutation extraction proposed in [LKC+18] uses several models to capture the relation of genes, mutations, diseases and drugs. With the focus on relations of the entities, the mutation normalization is rule-based. Word embeddings are used to increase the performance of the algorithm by getting contextual relationships of the terms.

The mutation extraction systems published before 2017 share that they are rule-based and overtaken by performance of machine-learning based processes in and after 2017. Common to all normalization approaches is a small gold dataset regarding normalization as well as a rule-based matching process.

This work can be best compared to tmVar2.0 [WPF+17], where the normalization process is rule-based. There exist two mappings of mutations to unique identifiers. The first is based on a database lookup and the second uses co-occurence in the text. The database consists of mutation names along with their preferred identifiers and is extracted from ClinVar [LLB+17] and dbSNP [SWK+01]. This co-occurrence is evaluated by several patterns that match a pattern like 'gene', 'mutation', 'identifier' in some distance and relate them to each other. This information is then curated based on the knowledge in the database.

A direct application of using word embeddings for mutations is used [KLKK18]. In this publication word embeddings are used to improve the understanding of the

differencees and dependencies between mutations.

With ClinVar and dbSNP beeing databases that do serve also synonyms and tmVar2.0 and SETH making use of it, no mutation synonym extraction system is known to the author.

2 Fundamentals

In this chapter, fundamental techniques and algorithms used in this thesis are explained. The given examples are based on section 1.3.

2.1 Named Entity Recognition (NER)

In Natural Language Processing (NLP) the task of Named Entity Recognition (NER) is the process of assigning named tags to find entities in a text. As a computer understands the text as a sequence of characters, neither the text nor the words have a meaning to it. Using defined rules, there have been rather strict approaches to extract certain entities. A different approach by asking for commonalities and using statistical models has been successfully applied in NLP tasks [MMS99]. In NER, machine learning methods are proposed to be the 'best independent solution' [MAM08].

Using the guiding example, a simplistic rule to extract mutations like "V600E" from text is: *a single capital letter from the dictionary of the amino acids, followed by a number of digits and trailed by another letter from the dictionary of amino acids*

Similarly, there have to be extensions to this rule, like allowing space in between and restrictions as to ignore the pattern if it occurs inside another word. Many systems for recognizing such entities are based on rules, while machine learning techniques typically outperform rule-based systems. A commonly used machine learning technique are Conditional Random Fields (CRFs) [LMP01].

2.2 Named Entity Normalization (NEN)

While NER is the task of assigning a concept to a word, Named Entity Normalization (NEN) is the task of linking common concepts to a single entity. These entities should preferably be normalized to unique database identifiers as proposed in [TKF+11]. Although these databases can be used for high precision mapping, they are not covering all different spellings of mutations actually used in the text.

Using the guiding example, the mutation "T1799A" should be recognized as being a mutation entity in NER and subsequently normalized with the mutation "V600E" to the unique identifier "rs113488022", following the information in [SWK+01].

The software TERMite from the company SciBite is used in this thesis to apply NER and NEN on other entities than mutations [Lim18] , see also subsection 5.1.3.

2.3 Text cleaning and simplification

In addition to NEN (see section 2.2) also the cleaning and simplification of text is important. One cleaning technique is the removal of stopwords. These stopwords are words that are note relevant for the understanding of the text and can be removed without changing the meaning of the text drastically. Examples of such words are 'and' or 'or'.

To simplify the text after NEN is applied (see section 2.2), the found entities can be replaced by the preferred label. This can reduce the complexity of a text. The meaning of the word is then not encoded in the context anymore but made explicit by the preferred label. With this also multiword entities can be reduced to a single word in the text. Further simplification can be applied to words not normalized by NEN: plural and singular forms can be reduced to the singular form, and the multiple forms of a verb can be reduced to the simple form. This process is called lemmatization.

2.4 Vector Space Model

In the supervised machine learning domain, features are extracted from the texts for the tasks NER (see section 2.1) and NEN (see section 2.2). These features can be very low level textual features like numbers occurring in a word but also high level features like the surrounding words.

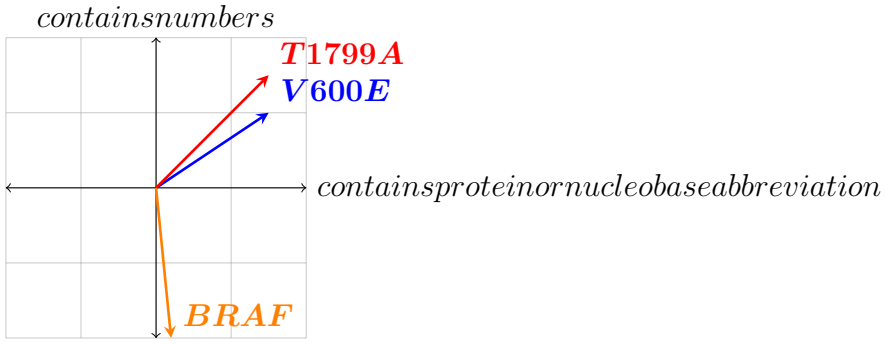
2.4.1 Vector encoding of words

The textual and contextual features can be understood as dimensions of the task [MMS99]. For the training, the classification vectors are built according to the rules of the features. For evaluation, the word that has to be tagged is placed in this n-dimensional field according to the features and the nearest training vector is searched. Looking at section 1.3, two exemplary features could be that the word contains numbers or that it contains protein or nucleobase abbreviation letters. For nucleobases this would be 'A', 'C', 'G', 'T'. Figure 2.1 shows three vectors. 'V600E' and 'T1799A' are words containing both, protein and nucleobase abbreviation letters. As the number of the abbreviation letters is the same, their 'abbreviation letter' axis part is the same. Also both are containing numbers inside, therefore both vectors contain also a part for the 'contains numbers' axis. The third vector 'BRAF' which is the gene name, does not contain any numbers and has a negative

amount on this axis. Containing one nucleobase letter ('A') there is a little positive amount in this axis. This is a false positive, as in this case the 'A' does not stand for a nucleobase.

Comparing the three vectors by the distance of their ending points, the vector 'V600E' is more similar to the vector 'T1799A' than to the vector 'BRAF'.

Figure 2.1: Vector illustration for tagging a word into categories.



2.4.2 word2vec

The example in Figure 2.1 gives an introduction to vector representations where the dimensions are rule based. In [MCCD13] two word embedding models are proposed that create a vector representation that preserves similarity of words based on the context they share. To achieve this, each word is transformed into an abstract vector representation in continuous space. In contrast to other vector encodings the dimensions do not refer to anything explicitly, and are initialized randomly. This is called the projection layer. It maps discrete entries (words) to a continuous space (vectors). Using this projection layer, for each word the surrounding words (the context) are predicted. Figure 2.2 shows the architecture of the Skip-gram model. What is not pictured is the weight matrix between the projection layer and the output layer. In using this weight matrix for each context word, a matrix of candidates is produced. Each of these context matrices consist of all words in the vocabulary. For each vocabulary word in each context matrix the weights and the vectors are evaluated. The output is a probability that the word from the vocabulary is the word that is expected at this contextual position. The loss function is calculated and the matrices are updated.

These models are shallow neural networks that learn an abstract encoding based on the prediction of context words. The probabilities of the context words are calculated as explained in [Mey16] and [GL14]. In the Skip-Gram model the probabilities that a word, that occurs at position $t - x$ before the word in focus w_t is the j-th word in the vocabulary, is denoted as $w_{t-x,j}$. In the sample $w_{O,t-x}$ is the actual occurring

word at position $t-x$. A softmax function is used to calculate the probability. This is the fraction of the probability to see this word at this position in the context of the word in focus (written down as $u_{t-x,j}$), against the sum of the predictions of all other words (denoted by $u_{j'}$) (see Equation 2.1).

$$P(w_{t-x,j} = w_{O,t-x}|w_t) = \frac{exp(u_{t-x,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.1}$$

In the model implementation the word to context probabilities are saved into two matrices. The first matrix has the dimension in number of words of the vocabulary times the number of features. These features are abstract features to learn (compared to the very explicit manual curated features from the example in Figure 2.1). This matrix is called the projection matrix. The other matrix has the number of features times the words as dimensions and is called the output matrix.

When using 2 abstract features with a vocabulary of 4 distinct words, the projection matrix P is then a 4×2 matrix and the output matrix O is of size 2×4. The prediction in the upper part of Equation 2.1: $exp(u_{t-x,j})$ can be expressed as a matrix multiplication of the projection matrix by the output matrix and selecting the appropriate rows. With the first word as the word in focus, the first row is selected and results in a vector of size 1×2 P_1 from the projection matrix. This vector is then multiplied with the output matrix 2×4 resulting in the 1×4 matrix R. This encodes then the sum of the value from the projection layer and the weight from the output layer to a value that resembles the probability of the word occurring in the context of the word in focus. With probabilities not summing up but multiplying, the logarithm of the probability is encoded into the features and weights and then summed up. The result is then potentiated.

$$P = \begin{bmatrix} p_{01} & p_{02} \\ p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix}$$

$$O = \begin{bmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \end{bmatrix}$$

$$P_1 = \begin{bmatrix} p_{11} & p_{12} \end{bmatrix}$$

$$R = \begin{bmatrix} p_{11}w_{10} + p_{12}w_{20} & p_{11}w_{11} + p_{12}w_{21} & p_{11}w_{12} + p_{12}w_{22} & p_{11}w_{13} + p_{12}w_{23} \end{bmatrix}$$

This approach is very compute intensive. In every calculation step the probabilities for each word is calculated, but this can be optimized. The softmax is not computed for each word individually (so everytime a one vs. all calculation) in the first time. But representing the output vocabularies as a binary tree where the inner nodes represent the probabilities to take the path to reach the node. The actual probability of seeing one word in the context (a leave) is then the product of the inner nodes on the route to the root. This reduces the calculation for each context word from the complete Vocabulary V to the depth of the tree, which is $log_2(V)$ [MH09].

This reduces the training complexity Q to Equation 2.2 like shown in [MCCD13]. C is the context size of words (in the window), D the number of dimensions and V the used vocabulary.

$$Q = C \times (D + D \times log_2(V)) \tag{2.2}$$

In contrast to the approaches in section 2.1 and section 2.2 word embeddings are learning the contexts of words. Therefore the word vectors can be described as *the word encoding is learned based on contextual similarity.* These contexts can be measured according to their similarity and the words producing similar contexts can be seen as synonyms. *A word pair is a synonym of each other if they can be exchanged without changing the meaning of the context.*

Word embeddings have been applied in different fields for entity recognition, normalization (e.g. in [CCKP16], [UBP17], [GCWL14], [SBSPM15] and [WXJ+15]) and synonym extraction ([SCO09]), but also in the mutation domain ([LKC+18] and [KLKK18]). Precomputed word embeddings exist for PubMed and PubMedCentral ([MA13]).

2.5 Knowledgebase construction

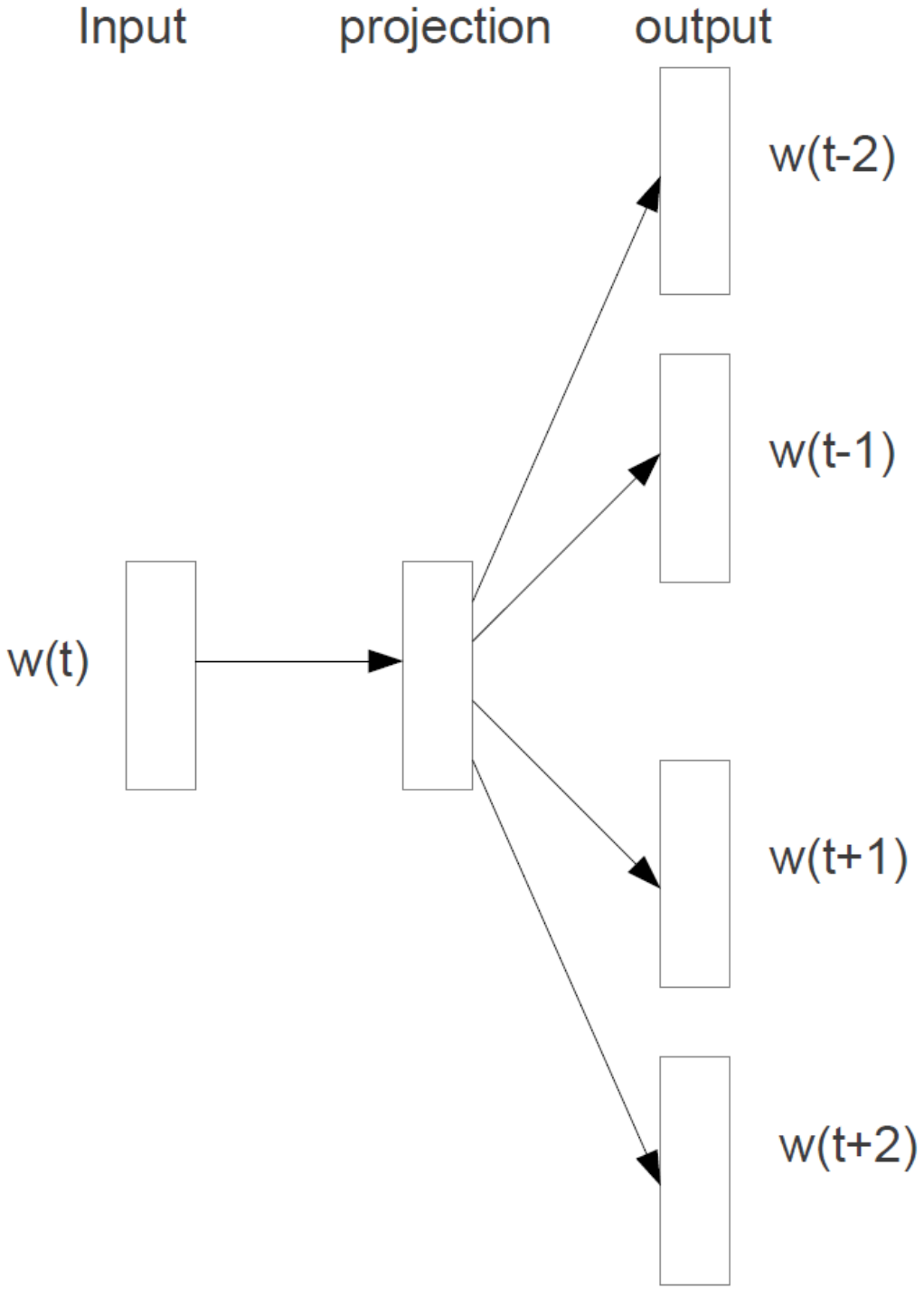
A knowledgebase (KB) contains entities, their relations and mentions. A KB containing genetic variants enables the user to get information about a mutation and about possible other names of the mutation occuring as synonyms. Also other systems are able to use the KB, as its data can be read in a relational format. This upstream system can then evaluate the linked entities that are returned by the KB. The KB can also serve as a starting point for creating an ontology of genetic variants as they occur in the literature.

2.6 Full text

PubMed contains abstract texts (see subsection 5.2.1). These abstracts are rather short in their nature and contain a condensed version of the fulltext article. In

[WST+17] and [VLBW+13] it is shown that extracting information from fulltext articles is consistently outperforming an approach using only abstracts. Especially to get enough data for genetic variants that are rarely occurring in text, fulltext sources need to be used.

Figure 2.2: The Skip-gram model architecture to create word embeddings presented in [MCCD13]. $w(t)$ represents the word w at position t. The projection is in the center. For every potsition in the context the output matrix is depicted.



3 Model creation

The application of the word embedding algorithm is presented in this chapter. As the preprocessing is important to the applicability of the algorithm, it is presented as well. Figure 3.1 contains an overview of the data flow and how the models are constructed.

3.1 Preparation of the data

3.1.1 Uploading the raw data to the cluster

The data from the sources has already been downloaded to a shared folder, and it is therefore not necessary to download the data from the providers again. In the case of PubMed and PubMedCentral the XML files are compressed together in containers (e.g. based on Journal Name). ScienceDirect is stored as individual xml files. As both formats do not fit optimal into the distributed filesystem of Hadoop (HDFS), a small python application is used to combine the XMLdata into a more Hadoop friendly format. Apache Parquet has been chosen as file format over textfiles. Parquet enables the use of block-wise compression, contains the schema by itself and allows fast reads and writes [Voh16]. The python application takes four parameters:

1. The input folder
2. The target directory path on HDFS
3. The WebHDFS gateway (server name and port of the user facing server)
4. The number of documents to combine in one parquet file

The application reads through the input folder 1. and creates a list of all files in the directory. While iterating through the parameter from 4. it creates batches. For each batch it creates a parquet file with a single column, where the raw xml content is put in. The folder structure from the source data and order of the files is not relevant. This parquet file is then uploaded through the WebHDFS gateway 3. to the specified target directory 2. With this small intermediate step, a good data distribution can be ensured and processes using this data later can benefit from distributed reading. Table 3.1 shows the schema of the file.

Row	$XMLcontent(String)$
1	$<?xmlversion = "1.0" encoding = "utf-8"?><article[...]$
2	$<?xmlversion = "1.0" encoding = "UTF-8"?><full-text[...]$
3	$[\cdots]$

Table 3.1: XML content of the data is converted to a parquet table.

3.1.2 Extracting the data from the XML files

As the data is now living in HDFS, all applications are multi-process applications that run on many machines in parallel. This advantage comes with additional effort in the fields of programming, debugging and optimization. To ensure the data is properly extracted, all necessary functions are implemented such that they can be tested locally.

A Spark parquet reader loads the data into the Spark environment, and represents it as one dataset. For each of the row elements in this dataset an XML extraction function is called. This XML content is then fed into the function as a string. The python built-in XML parser ETree is used to extract the data from the String content and to return this information in a python dictionary. This python dictionary can be directly used by Spark dataset tools to read it back to a nested table format. This nested table format now contains the relevant columns like the title, abstract and if available the full text of the article next to the metadata fields, like the publish date, the corpus it was originated from and identifiers that can be used for deduplication (e.g. PubMed Identifier).

All data sources are containing rich markups inside the text, for example section titles, tables or an italic font. After the removement of these markups, the text is extracted. Additionally also fields for identifiers are queried like the pmid, DOI (Digital Object Identifier) or the journal name. The final layout of the data is shown in Table 3.2.

3.1.3 Deduplication of the data

PMC data as well as the data from ScienceDirect contain PubMed identifiers (pmid). This identifier is used to deduplicate the data. Deduplication of the data is necessary as there is an overlap of abstracts from PubMed with content from the additional full-text sources. A duplicate sentence would imbalance the training set for the word embeddings. In the majority of the articles the deduplication of the content is straightforward. Most articles from the full-text corpora contain the PubMed identifier to which the full-text can be linked directly. By this process many articles from PMC can be linked to Pubmed. For the ScienceDirect corpus many elements have not been able to be deduplicated by using the pmid. The articles that cannot

source	pmid	doi	pmc-id	journal	date	article title	abstract	body	full text
PMC	29426306		5807846	BMC Cancer		Characteristics of [...]	[...]	[...]	Characteristics of [...]

Table 3.2: XML content of the data is converted to a parquet table.

be linked are appended to the corpus. A common text field is created. This field contains the whole text of the article: title, abstract and all body sections.

As the dataset combines a many short abstracts as well as long full texts, the data is skewed. Half of the articles are very short with 968 characters or less. While there are comparably few with more than 33241 characters per article (see Figure 3.2).

3.1.4 Tokenization of the text

For the simple model text is tokenized. In this process the whitespace (or similar characters like tabs or newlines) is used to split the text into a sequence of words. Splitting a sentence by any typical end of line symbol does not work, as the full stop is contained in mutation expressions as well (eg. in 'c.1799T>A'). Therefore the full stop must not be used when characters or digits follow directly. The words are then lowercased. An example sentence of:

'BRAF is associated with the indication melanoma and the mutation c.1799T>A.'

would be tokenized and lowercased to the form:

'braf, 'is', 'associated', 'with', 'the', 'indication', 'melanoma', 'and', 'the', 'mutation', 'c.1799t>a'

This list of words is representing just the lowercased tokenized words without any alteration of the order or content of words. All words are contained as they occur in the text.

3.1.5 Apply TERMite and Stanford CoreNLP

In the advanced model the software TERMite from SciBite is used to normalize entities in the text to a common label. After the deduplication and before the model construction, a NER and NEN step is introduced to replace different names for an entity to a common label. Additionally a step using Stanford CoreNLP is used to split the text into tokens, remove stopwords and lemmatize words [MSB+14]. With stopword removal the filling words 'is', 'not', 'with', 'but' and similar are removed. By lemmatizing for example the word 'associated' is converted to its base form 'associate'. The order of the words is preserved.

With TERMite and Stanford CoreNLP the sentence:

'BRAF is not associated with non small cell lung carcinoma, but with c.1799A>T and V600E.'

is tokenized and normalized to the array: 'b-raf_proto-oncogene,_serine/threonine_kinase', 'associate', 'non-small_cell_lung_cancer', 'c.1799a>t', 'v600e'

3.2 Model creation

Using the Apache Spark MLlib word2Vec implementation, the model is trained on the arrays that are produced on either the simple tokenization or the cleansing process (through TERMite and StanfordNLP). The output is the projection layer of the word embedding model that assigns for each word in the vocabulary a multidimensional vector. Similar words, based on their context, will have similar vectors.

3.3 Knowledgebase creation

The mutations and their synonymous names need to be retrieved from the model in a user friendly way. Therefore a web interface, where a user can enter the word for that the synonyms are displayed, can be used. Additionally the similarity of the input word to each of the synonyms is displayed.

Figure 3.1: The data flow inside the application.

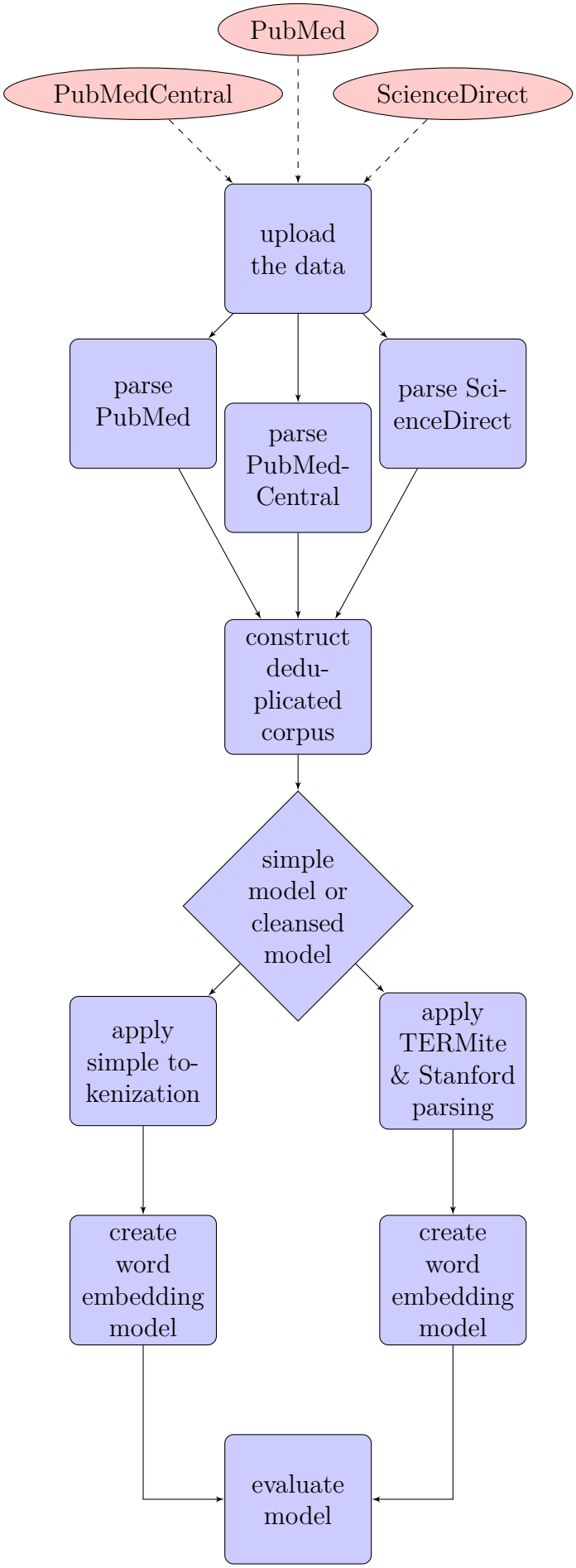
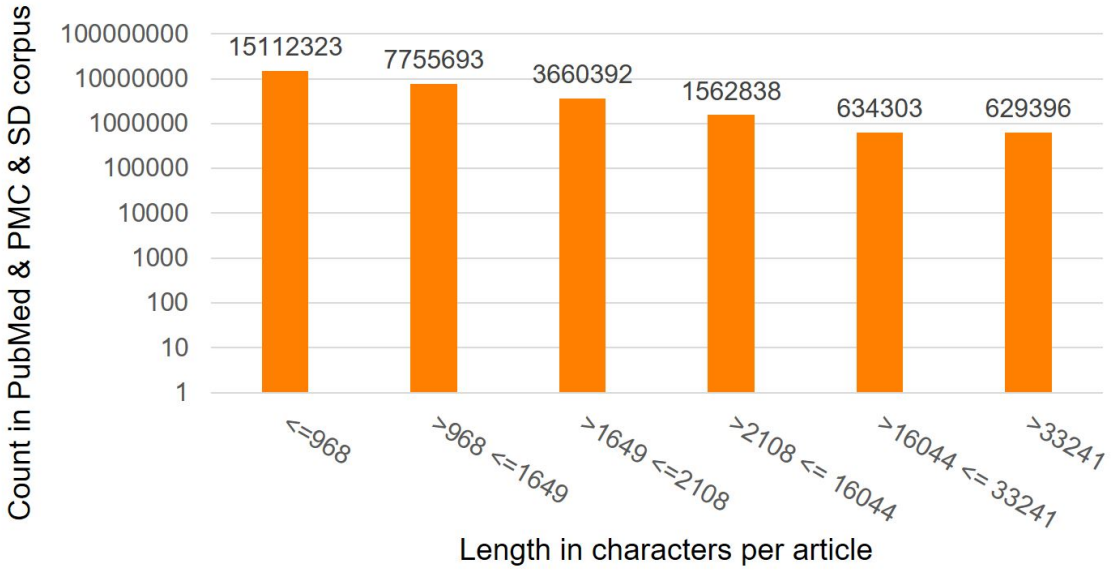


Figure 3.2: The data is skewed, as many short articles are combined with relative rare long full text articles. The columns are based on the median on the length of the articles. The majority of the articles (the first thee columns) contains shorter abstracs with less than 2108 characters per article.



4 Information Extraction

The information about mutation mentions, their relations and how they are extracted from the raw text is described in this chapter.

4.1 Named Entity Recognition

Named Entity Recognition in the biomedical domain is often performed using Conditional Random Fields like in [BDS+08], but also dictionary approaches exist as used in [PSP+06]. In [TSV+12] a combination of various tools is used to tag the entities and create links between them. For the recognition of mutations, a rule based approach makes sense at the first glance: The nucleobases are known and are limited, as well as amino acids. Additionally normalization is possible using standardized databases like [LLB+17] or [SWK+01]. Benefitial of these standardized databases is, that these are typically manual curated databases that also follow strong nomenclatures. This indicates a low false positive rate. Otherwise, these strong nomenclatures are probably not always reflecting how entities are written down in a publication. A prominent example from section 1.3 is 'T1799A', that occurs in text (the PubMed search interface returns 86 results on 2018-0-06). It does not follow a standard nomenclature and is hereby also not contained in the evaluation set (see section 5.4).

Therefore an entity extraction that detects mutation mentions similar to a standardized entity naming is neccessary.

4.2 Word Embeddings

The word embedding model (described in subsection 2.4.2 and explained in detail in [Ron14]) is based on the principle that words producing a similar context share a similar vector encoding (in the projection layer). This principle is used to extract mutation mentions from the text only based on contextual co-occurence.

Based on the guiding example (see section 1.3), and using the two queries presented in the same section next to a fictive example, three exemplary training samples can be constructed:

1. '[...] presence of the BRAF V600E mutation were evaluated [...]' (excerpt from [AMAS18])

2. '[...] frequency of the BRAF 1799T>A mutation in Mexican [...]' (excerpt from [FRHLL+18])
3. '[...] synonyms from the BRAF 1799T>A mutation have been [...]' (fictive example)

These three sentences are transformed through tokenization and lowercasing into a list of words:

1. [...], 'presence', 'of', 'the', 'braf', 'v600e', 'mutation', 'were', 'evaluated', [...]
2. [...], 'frequency', 'of', 'the', 'braf', '1799t>a', 'mutation', 'in', 'mexican', [...]
3. [...], 'synonyms', 'from', 'the', 'braf', '1799t>a', 'mutation', 'have', 'been', [...]

The word embedding model predicts the surrounding words for the word in focus. Table 4.1 shows the words for that the model is training with the word '1799t>a' in focus. Only sentence 2 and 3 are selected as training samples, as sentence 1 does not contain '1799t>a'.

sentence	w(t-3)	w(t-2)	w(t-1)	w(t)	w(t+1)	w(t+2)	w(t+3)
2	'of'	'the'	'braf'	'1799t>a'	'mutation'	'in'	'mexican'
3	'from'	'the'	'braf'	'1799t>a'	'mutation'	'have'	'been'

Table 4.1: Word samples for the words in focus

Table 4.1 can be read in the following way:

- The word w(t) is the word that is used for the prediction.
- The context window is 3; therefore 3 words before and after the current word are predicted.
- For each training sample the probability that word w is occurring at offset x from the current word is calculated.

Using the training words from Table 4.1 and the equation from subsection 2.4.2 the likelihood of seeing each word at its offset is maximized.

$$P(w(t-1) =' braf' | w(t) =' 1799t > a') = \frac{exp(u_{t-1,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{4.1}$$

In Equation 4.1 the likelihood of the word 'braf' directly preceding the word in focus '1799t>a' is evaluated against all other words. The probability is calculated for 'braf' against the probability of all other words. j is the position of the word 'braf' in the vocabulary V. u is the prediction value from the projection layer multiplied with the output layer.

With many additional training samples (sentences in the corpus containing 'v600e' or '1799t>a'), the model updates the probabilities in the projection and weight matrix appropriately. In the case of 'braf' as the current word and the context size 1, the word 'the' at position w(t-1) would get a higher probability than the words 'v600e' or '1799t>a'. The probabilities are encoded into the matrix multiplication of the projection matrix and the weight matrix (see subsection 2.4.2). Many contexts are not the same as in the constructed example and even in this example the items 1 and 2 share many words. With a larger context more words are different than shared. Therefore it is essential to not explicitly compute the probabilities for an individual word but to reduce the amount of features (not words but abstract dimensions) to create inference. Using these abstract features with the weights, word embeddings create a fuzzy matching of words to contexts.

Word embeddings are very sensitive to word boundaries as it operates on single words. In case the full stop would be treated as a token to separate sentences or words, the 'c.1799t>a' and the 'p.v600e' would be split by the full stop. This needs to be avoided, as these parts need to stay together for the synonym normalization. Therefore, the basic NLP tasks like lemmatization or stopword removalcan help in this context or erase valuable information. In [FFJ+16] it is proposed to use lemmatization to simplify the input text for a synonym generator.

4.3 Similarity

The similarity of two words w_p and w_q can be measured by their vector representations $\vec{p}$ and $\vec{q}$. There exist two different measurements:

- Euclidian distance : the distance is measured by the square root of the sum of square differences of each component for each vector. $d(\vec{p}, \vec{q}) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2}$
- Cosine similarity : the cosinus of the angle between the vectors compared to their length. This is then: $sim(\vec{p}, \vec{q}) = \frac{\vec{p} \cdot \vec{q}}{||\vec{p}||||\vec{q}||}$. In case of normalized vectors this can be computed with the dot product directly ([MMS99]).

The more similar the contexts are, the more similar are the vector representations of the words. This leads to a low distance or high similarity score. Both methods produce the same ranking, therefore the cosine similarity is used.

5 Implementation

The used tools to implement the extraction of the data up to the evaluation of the model are presented in the following.

5.1 Tools

Various tools, programming languages as well as execution frameworks are used to implement word embeddings from the raw publication data to the web interface for querying.

5.1.1 python and libraries (ETree)

The programming language python is chosen as default implementation language. The main reasons are availability of the modules for handling the data, interfaces to the used frameworks and reusability of the code. For extracting the content from the XML to the content fields the module lxml is used (see subsection 3.1.2). The benefits of using lxml are that it features an iterative parser for reading the data in blocks, understands namespaces and gives the possibility to remove certain tags. Iterative parsing is important for the very long documents occurring in the ScienceDirect subcorpus. Removing certain tags is used in PubMedCentral and ScienceDirect data as both contain rich markups in the text (e.g. italics, tables or figures).

5.1.2 Apache Parquet, Hadoop and Spark

As implementation platform Apache Hadoop is used. This platform enables a data parallel processing for applying the extraction from the xml (see subsection 5.1.1), as well as the deduplication and corpus generation (see subsection 3.1.3), the tokenization (see subsection 3.1.4), the distribution of the text simplification through SciBite TERMite (see subsection 3.1.5) and the word embedding model training (see subsection 2.4.2).

As the source data is not substantially large, even commodity computers can handle the data volume. But executing the processing on parts of the data in parallel can reduce the overall runtime.

In case of the text simplification and tokenization even with running at 50.000 words/second/core, for the nearly 12 billion words in the corpus over 66 hours would be needed on a single core. As the text annotation and replacement is an extensive task, the speed drops to around 5.000 words/second/core. Parallelization on 30 cores with individual models runs in a day.

Training the word embedding model (subsection 2.4.2) on a single machine would require a moderate amount of memory but a substantial amount of time. The projection and output matrix, each with the dimension $V \times D$ (with V as the size of the vocabulary, D as the number of dimensions), as well as the tree for the hierarchical softmax need to be kept in memory. Also in [MCCD13] a distributed approach is used for an even smaller data set with 1.6 billion words in total.

In all stages Apache Parquet is the chosen format for the data. It can be seen as a data container that allows to store the data in a columnar format (in contrast to a row oriented format). The container supports commonly used datatypes. The built-in compression can reduce the size of the data and reduce the time to read the data. Parquet is well integrated into the Hadoop framework such that most tools support the file format natively. Also the python module pyarrow supports parquet files and therefore enables to upload the data as Parquet files directly.

Apache Hadoop also supports Parquet files directly. The Hadoop ecosystem contains several different frameworks for storing and executing tasks on many different computers in parallel [SKRC10]. The foundational part of storing data across multiple computers is called the Hadoop Distributed Filesystem (HDFS). HDFS handles computer failures in software and allows to store data larger than a single commodity computer. The filesystem is exposed as it would reside on a single machine. Therefore a developer does not need to know on which computer which file part is stored.

Executing tasks on this data stored in Hadoop is possible with different tools in its ecosystem. One of the tools is Apache Spark which does not necessarily need to run on Hadoop but integrates well into it. Running on Hadoop, Spark reduces the exchange of data over the network by executing tasks on the computers, where Hadoop has stored the part of the data. Spark incorporates several libraries, from that Spark SQL and MLlib are used. The SQL library allows to use query expressions based on a table like format of the data. It reads/writes parquet data natively. The query expressions look familiar to SQL function used in relational databases.

In addition, it is possible to extend the functionality of Spark by User Defined Functions (UDFs). With UDFs other functions, like a non Spark python function can be included in the workflow and then applied to a column. In the Spark MLlib there exists a word2vec implementation for training word embeddings using the Skip-Gram model ([ZCF+10]). There exist other implementations for running word2vec on a very large corpus ([SSWT17]). As the corpus was not that large in size, the good integrated solution from MLlib is used.

5.1.3 SciBite TERMite

The company SciBite offers the software TERMite for doing Named Entity Recognition and Named Entity Normalization on various datasources [Lim18]. The key components of the software are called Vocabs. These Vocabs (from now on called vocabularies) can be understood as taxonomies including exclusion rules to minimize overlap in the tagging. The server component allows tagging via a REST API, a batch process, a web interface as well as an embedded Java application called TERMiteJ. With the license of TERMite that was granted for this thesis, vocabularies are included. These vocabularies enable matching entities that occur in the biomedical context of the coprora. These entities include: genes, drugs, indications, chemicals and many more.

The TERMite vocabularies also offer a vocabulary for tagging dbSNP identifiers and mutations. But with these vocabularies no normalization of the recognized variants is performed. With TERMite it is possible to use a custom build vocabulary, and to apply this vocabulary for NER/NEN on other data.

For running it on top of the substantial amount of text, TERMiteJ is included into a scala User Defined Function (UDF). In this UDF not only the text is annotated but also the text is replaced by its preferred label. For ambiguous annotations, always the first proposed label is used.

In cases were TERMite is not able to assign any label to the text, the actual text content is used. Therefore it is expected that the number of tokens can grow.

The run of TERMite for annotating the text with 37 dictionaries in the distributed environment is done in less than 12 hours in a shared distributed environment on nearly 12 billion words (including reading and writing from disk, replacing with the peferred labels and Stanford CoreNLP stopword removal and lemmatization).

5.1.4 Stanford CoreNLP

The Stanford CoreNLP ([MSB+14]) toolkit is used to simplify the text even further. One of the simple tasks is to remove constant stopwords. These stopwords are words like 'and' or 'or'. This can be done using a simple blacklist approach. This enables the word embedding model to include more important words without increasing the window size. Another important step performed is the lemmatization of words. This removes the inflected forms of a word and keeps only the lemma of the word. For example the words 'good' and 'better' would be two distinct words without the lemmatization. With it, 'better' would be replaced by 'good'.

5.1.5 N-Gram creation

Before training the embedding model, bi-, tri- and quad-grams on words have been computed for the PubMed and PMC data. This step was excluded from the final

workflow as some mutation mentions (anyway being rare) are preceded by the gene, and succeeded by a rs identifier. This coincidence is use in [WPF+17] to get more information about mutation mentions. The multiword construction is based on the algorithm used in [MCCD13]. The algorithm is counting the co-occurence of word pairs. In case a word is often occurring together these words are joined together whenever they occur together. This is done down to a certain threshold. By executing this procedure two times and feed the output from the first in the input of the second round, also tri- and quad-grams are constructed.

By using this simple method many interesting multi-word entities can be reconstructed that have been separated by a space character before. For example 'amino acid' occurs nearly 620,000 times in the whole PubMed corpus together, while 'amino' without 'acid' only around 70,000 times. But also publishers ('john wiley sons ltd') or gene names ('spiel ohne grenzen') can be reconstructed using this algorithm. The runtime of the two rounds is around 47 hours on 40 cores with a total memory of 460GB. This computation runs on a total of 28,031,441 entries, with a compressed size of 34.4 GB on disk. With this amount of memory it is possible to hold the complete Dataset of over 27 million abstracts of PubMed with the fulltexts of the 1.7 million PubMedCentral in memory as well as the tables for the bi-gram and the tri- and quad-gram exchange.

5.1.6 WordEmbeddings

The word embedding implementation that is included in Apache Spark MLlib is used. It provides the Skip-Gram model architecture and runs in the distributed environment. The word2vec model is executed on several replicas of the matrices that get merged after a training phase. As the training data (the text) is split over multiple computers each computer trains a model on the data it has on its own disks and the models later get merged. Having many partitions of the data and only one update step much information will get lost. Therefore the merge of the model parameters is done several times (here: 5 times).

5.2 Data

The total raw data size is 100 GB zip compressed. With the upload of the XML content these files are unzipped and the content is uploaded as a compressed parquet file to HDFS.

5.2.1 PubMed

The National Institute of Health (NIH) offers with PubMed one of the largest corpora of scientific abstracts that are free to download and use in applications. The

corpus is offered in a semi-structured XMl format which contains metadata of the publication in a structured format (for example the PubMed ID, DOI, authors, and more), as well as the abstract text inside a field. The corpus currently contains about 27 million entries and has a download size of nearly 30 GB. The documents in focus are mainly publications in medicine, pharmaceuticals and life sciences topics. Furthermore there exists a minor amount of publications in other fields such as chemistry or physics. In addition to this corpus the NIH offers a web search interface for querying the corpus, as well as a suite of tools that are used to tag different types of entities like chemicals, genes, and recently also mutations. The system used to tag mutations is called tmVar ([WPF+17]). The application of these tools is offered in PubTator, an annotated version of PubMed, which is usable through a public web search application and for download.

5.2.2 PubMedCentral

In addition to the abstract corpus, the NIH also offers an additional corpus containing full-text articles for the articles in PubMed. A subset of this corpus, the OpenAccesss subset can be used for machine learning tasks. The NIH is offering the full corpus through a search interface as well as download. Currently the full corpus contains about 5 million full-text articles of which more than 1.7 million are present in the OpenAccess subset. The data is as well provided in XML format, and contains additionally extensive text-markups (e.g. XML tags for italics or cross references). In the structured part of the text there exists for most of the articles also the corresponding PubMed identifier to simplify the linkage and deduplication between the two datasets.

5.2.3 ScienceDirect by Elsevier

The publisher Elsevier is offering full-text articles based on a license for individual retrieval, based on searches. The sub-corpus contains articles related to genes, mutations and diseases. The dataset that is used has a size of about 64 GB and contains approximately 2 million full-text articles. These articles are focused on the pharmaceutical, medical and life sciences domain and contain only a few articles from other domains. The xml format has more content than PubMedCentral.

5.3 Runtime and Hardware

As the data size and the amount of computations on it is at the border of being a good fit for a distributed architecture, the pipeline can use the parallel execution on multiple cores (see subsection 5.1.2). For all processes a shared Apache Hadoop cluster was used. This cluster consists in total of 12 worker nodes with a usable 725

GB Memory and 192 CPU cores. The machines are shared with other processes of the Hadoop environment, the memory and the CPUs are dedicated to the processes. The cluster is also shared with other tasks, so the runtimes can fluctuate a bit. All data is first transferred to the distributed filesystem, and all processes are started in the environment. Apache Spark is used from python as a framework for the distributed computation. Although the complete size of the data is not as large to be a perfect fit for a distributed setting, using a substantial amount of memory for using distributedXMLxml conversion and extraction as well as for building the model enables reasonable runtimes for a more interactive workflow.

- PubMed (xml conversion): For the conversion of PubMed XMLs to a tabular format the databricks XML parser has been used to extract 27,837,540 articles. The extraction took an hour to complete on shared hardware.
- PubMedCentral (XML conversion): As the databricks XML parser is failing at the deeply nested and open XML markups of PubMedCentral files, an XML extractor based on ETree has been developed. This extractor directly extracts the necessary fields and removes all markups that are not relevant. With this 1,863,349 articles are extracted. Of these articles 134,915 have no PubMed identifier. The storage size is nearly the same as for PubMed as the articles contain the full text.
- Science Direct conversion: For the extraction of ScienceDirect the XML parser had to be extended to iteratevely parse the documents. Some of the documents contain very much text. The ScienceDirect XMLs contain many markups like tables or references that are removed while parsing. From the ScienceDirect subcorpus 1,837,109 articles are extracted.

5.3.1 Deduplicating PubMed and PMC

The deduplication of the data is performed using the PubMed Identifier that was extracted from the XMLs. If there is no deduplication partner found in PubMed then a new entry is added. An overview about the overlapping of the data is given in Figure 5.1.

5.3.2 Model preprocessing

For each entry in the deduplicated data depicted in Figure 5.1, either a simple tokenization is called or the extensive simplification.

The simple tokenization runs in less than half an hour. For PubMed and PubMedCentral this results in 11,227,566 distinct and 10,804,447,233 total words. These numbers are in line with the reported corpus statistics in [CCKP16].

The TERMite and Stanford CoreNLP normalization and cleansing takes less than 12 hours on the PubMed, PMC and ScienceDirect corpus.

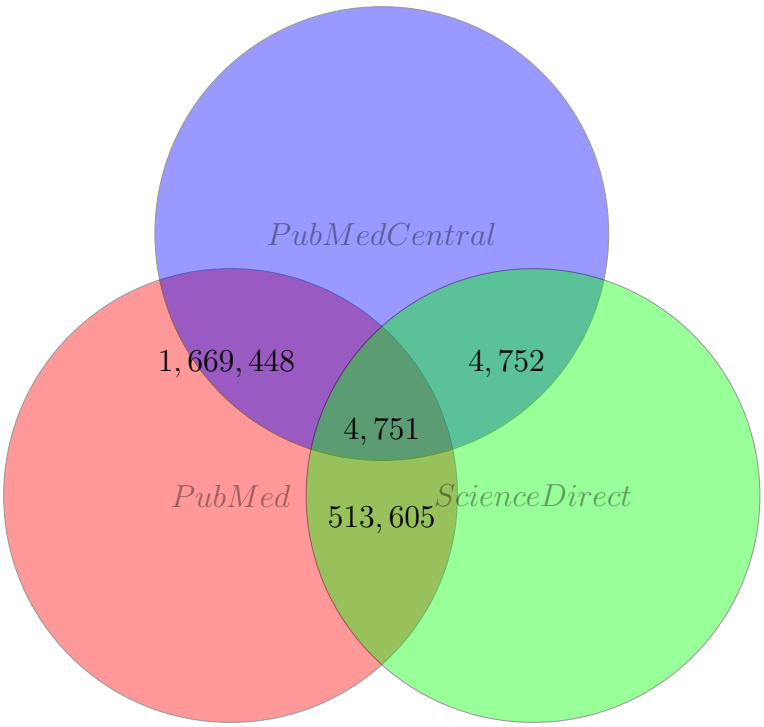


Figure 5.1: A visual representation of the overlaps of the articles in PubMed, PubMedCentral and the ScienceDirect subcorpus based on the PubMed identifier.

5.3.3 Word Embedding construction

In [CCKP16] hyperparameters based on PubMed, PubMedCentral and the combination together have been evaluated in detail. Based on their results, the Skip-Gram model is outperforming the CBOW model in its similarity benchmark by more than 10%. Also in [MCCD13] and [MSC+13] lowercasing the words is improving the similarity, by reducing the number of tokens. For the context window size, 20 was reported to have the hightest score in intrinsic evaluation, but with a window size of 1 leading to the best results in the extrinsic evaluation. In [LGD15] the window size showed a small effect on the intrinsic evaluation. The window size in [LGD15] and [CCKP16] indicate only minor differences between 5 and 10 words with similar performance in the extrinsic ([CCKP16]) and the intrinsic ([LGD15]) performance. Using the results from [MSC+13], the default context window size of 5 is chosen.

The reported number of dimensions having a good performance in the intrinsic evaluation on the similarity benchmark is 400, while for the semantic relatedness benchmark 500 performs slightly better. But on the extrinsic evaluation 200 dimensions have performed best. Additionally it is indicated that there is a huge increase from very low dimensions (25) and plateauing after 100 dimensions.

For the minimum count the highest value tested (2400) worked well in the intrinsic evaluation while in the extrinsic a middle-range number (50) performed better

[LGD15]. For the minimum count, domain specific effects are assumed as the evaluation data in the intrinsic evaluation is scoring the word pairs (test word pair similarity vs. cosine similarity inside the model), with ignoring words that are not contained in the test set. The intrinsic evaluation data used in their experiments contains rather often appearing names of drugs, symptoms and disorders ([PMA+10]). Setting the minimum count to a high number, only pairs with a high number of occurrence in the corpus are tested. For including also the rare mentions of mutations that are in focus, the minimum count is set to 20.

The vector dimensions encode the different semantic concepts contained in the text [SUY+17]. With the three large categories in the corpus, the rather medium amount of dimensions is appropriate for the evaluation in [CCKP16]. In the context of mutations the number of dimensions is set to the well performing 400 dimensions. Using the hyperparameter benchmark results from [CCKP16] and [LGD15] with the adaptions discussed before, the parameters that are used for constructing the models are shown in Table 5.1.

Parameter	Value
Model type	Skip Gram
Vector dimensionality	400
Min count	20
Window size	5
Learning rate	0.05
Lowercasing	Yes

Table 5.1: Initial hyperparameters for the word embedding models.

For constructing the word embedding model, the Apache Spark MLlib WordEmbedding model is used.

For constructing word embeddings the runtime based on PubMed and PubMedCentral data is: 12 hours on 100 cores with 490 GB memory and results in an embedding model that contains 1,250,325 words in the wordlist. The time is measured on shared hardware. Therefore the amount of cores that are available to the software is fluctuating. Table 5.2 shows that clearly on the cleansed model that runs in less than half of the time when the cluster is not occupied. The distinct words after the cleansing increase as not all words can be cleansed and therefore both versions of a word are kept. This is also valid for the lemmatization. Additionaly due to a limitation in Spark MLlib, it is necessary to increase the minimum count.

5.4 Evaluation data

The ClinVar data is downloaded and joined together. The final test data is converted to a tabular format for later usage. Table 5.3 shows the test example for the V600E

Parameter	Simple Model PubMed-PMC	Simple Model PubMed-PMC-SD	Cleansed Model
Pubmed	True	True	True
PMC	True	True	True
ScienceDirect	False	True	True
Cleansing	None	None	TERMite and Stanford
Documents	28,031,441	29,354,945	29,354,945
Words	10,804,447,233	11,936,304,678	11,936,304,678
Distinct words	11,227,566	18,790,878	18,790,878
Distinct words after cleansing	11,227,566	18,790,878	52,057,405
Minimum count	20	25	40
Distinct words in model	1,250,325	1,317,892	1,398,581
Model construction runtime	12 hours	11 hours	5 hours

Table 5.2: Word statistics of the contents of the models.

mutation of section 1.3.

gene	prefered label	synonyms
'BRAF'	'V600E'	['c.1799T>A', 'p.Val600Glu', 'rs113488022', 'Val600Glu', 'p.V600E', '1799T>A']

Table 5.3: Example for the evaluation data.

In the intrinsic evaluation data there are 1,422,369 synonyms for labels. Looking at the number of words that are contained in the model not all synonyms can occur (see Table 5.2). Compared to the number of total distinct words it also seems to be a fairly high number (12,67 % of all words would then be mutations mentions). Therefore the test set is reduced to the amount of words actually occurring in the corpus (see Table 5.4).

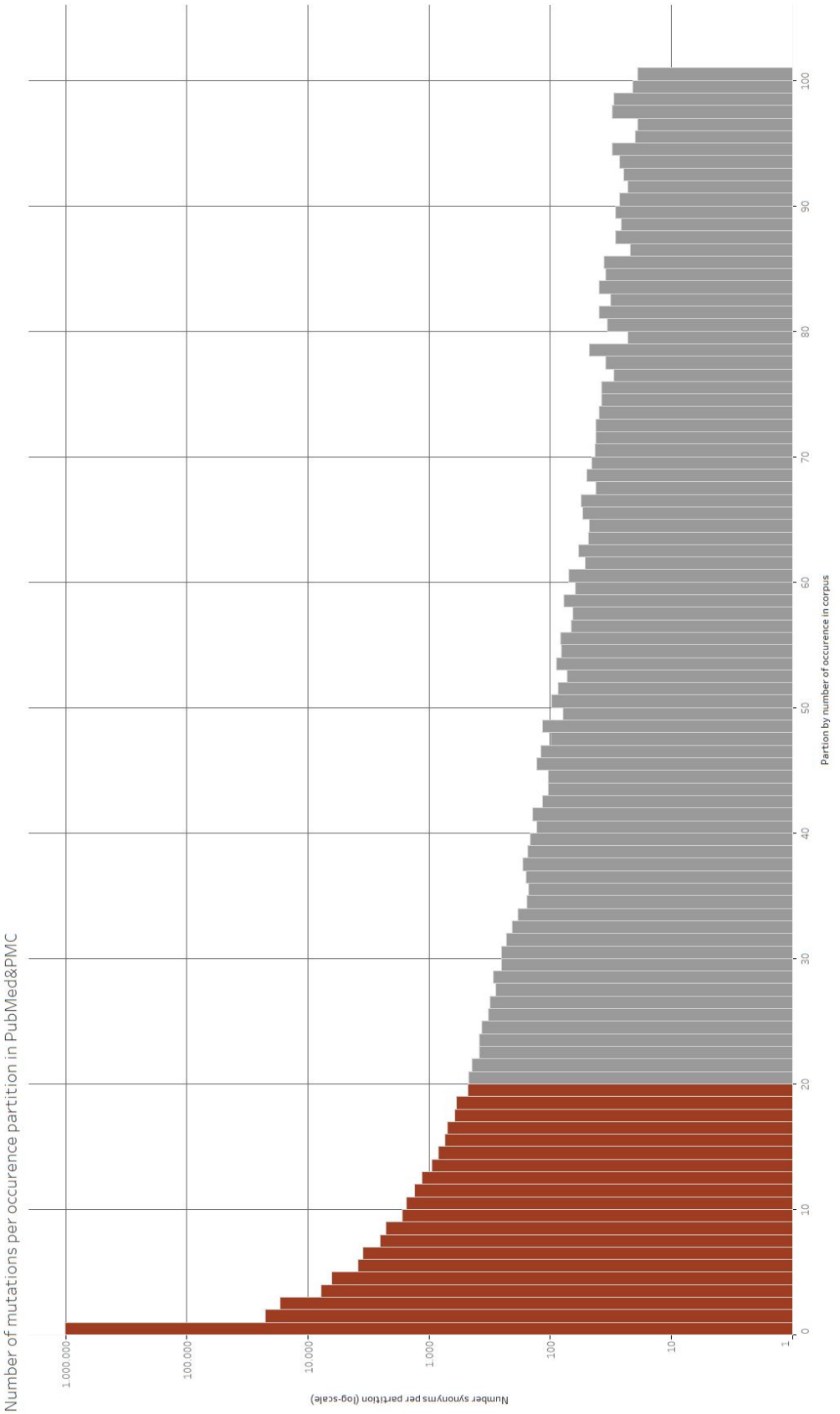


Figure 5.2: The number of mutation synonyms that are used in the evaluation set by the occurrence in the actual corpus. The ones that are not in the model due to the minimum occurrence parameter are red, the ones that are present in the model are grey. The chart is cropped at 100.

6 Results

6.1 Evaluation

As this model is not used for entity tagging, the model is not compared to other entity taggers. With the dataset from [TKF+11] also tagging entities are mandatory. Therefore an intrinsic evaluation is performed. The intrinsic evaluation is the evaluation of the model itself and whether it contains the key concepts [CFL13]. These key concepts are mutation mentions and their similarity, regarding to meaning the same mutation but on a different level (e.g. 'T1799A' and 'V600E'). With finding the closest points in the word embedding model in regards to cosine similarity, a open ended list is created. Through the nature of the problem, multiple correct answers for a mutation mention exist; searching for "V600E" from section 1.3 should output as well "p.V600E" as a close point. An absolute ranking is not applicable but a relative with common names that are closer to the input name. Measuring performance in precision and recall is performed with only the top n.

Here ClinVar is used as a source for genetic variants in humans. ClinVar is also maintained by the NIH and contains "more than half a million submitted records" ([LLB+17]). With the downloaded data from ClinVar, a table with preferred labels and their synonyms is created. Following section 1.3 the entry for 'V600E' is looking like:

- p.V600E
- Val600Glu
- p.Val600Glu
- c.1799T>A
- 1799T>A
- rs113488022

The model extracts synonyms from the text and does not tag mutations mentions. Therefore it is not comparable to other mutation taggers. Comparing the extracted synonyms with he manual curated data from ClinVar is applicable. ClinVar can be treated as a subset of a gold standard.

The evaluation allows to ask the question of similarity in two directions:

1. Querying the model for the preferred label and matching whether the synonyms are occurring in the top N results.

Model	Number of words in evaluation set
Unfiltered evaluation set	1,102,973
PubMed & PMC model	10,513
PubMed & PMC & ScienceDirect	8,935
PubMed & PMC & ScienceDirect Cleansed	2985

Table 6.1: Number of words in the evaluation set per model.

2. Querying the model for any synonym and matching whether the preferred label fo each synonym is occurring in the top N results.

In case a word is not present in the model it shall not be rated as a miss but is excluded from the evaluation. For querying the models, the evaluation dataset is filtered, whether the evaluation word is contained in the model. The resulting words for the evaluation are shown in Table 6.1. The lower number of words in the evaluation sets for the larger corpora is due to a higher minimum count value.

With these tests the results from Table 6.2 are obtained.

The subset contains only data where at least one match is found in the whole data. It is only evaluating on data that the model has actually seen. This evaluation is asking for the symbol that is the preferred label and then checks, whether the synonyms are contained in the resultset.

6.2 Error Analysis

The majority of the words is not occurring in the corpus (for example from Table 6.6). Many rare occurring mentions are not considered due to the minimum count (Table 5.4). Otherwise even for those mutation mentions occurring in the text and being contained in the model, the evaluation shows poor results.

The model shows better results results for queries that ask for a synonym and not the preferred label (compare Table 6.4 to Table 6.5). For the synonym > label query the expected label is ranked second, while in case 'v600e' is queried then 'c.1799t>a' is occurring at position 228. This is possibly because many other words sharing more similar contexts with the word than actually the other mutation mentions.

The model shows better results for queries that ask for a synonym and not the preferred label (comparing for example Table 6.4 to Table 6.5).

The initial baseline model using all word occurring in the model results in poor performance for the extraction of mutations. With the minimum occurrence of the words being set at 20, many rare mutation mentions are not contained in the model (see Table 6.6). While otherwise a token like "etc." occurs 72294 times in the text.

Model	Question direction	numTests	Precision K1	K1	Recall K2	K2
PubMed & PMC model	Label > Syn	1388	00.79%	1	05.18%	117
PubMed & PMC model	Syn > Label	1308	01.76%	1	10.10%	118
PubMed & PMC & ScienceDirect	Label > Syn	1055	00.90%	2	05.98%	117
PubMed & PMC & ScienceDirect	Syn > Label	1041	01.63%	1	11.71%	120
PubMed & PMC & ScienceDirect Cleansed	Label > Syn	217	03.23%	1	15.67%	107
PubMed & PMC & ScienceDirect Cleansed	Syn > Label	287	02.32%	3	23.88%	119

Table 6.2: Number of words in the evaluation set per model.

Model	Question direction	numTests	Precision K1	K1	Recall K2	K2
PubMed & PMC model	Label > Syn	99	11.11%	1	72.69%	117
PubMed & PMC model	Syn > Label	143	16.08%	1	92.40%	118
PubMed & PMC & ScienceDirect	Label > Syn	86	11.05%	1	73.33%	120
PubMed & PMC & ScienceDirect	Syn > Label	125	13.60%	1	97.52%	120
PubMed & PMC & ScienceDirect Cleansed	Label > Syn	42	16.67%	1	80.95%	107
PubMed & PMC & ScienceDirect Cleansed	Syn > Label	71	09.67%	1	96.53%	119

Table 6.3: Number of words in the evaluation set per model, when counting the result if there is at least one correct test.

Word	Cosine similarity
p.v600e	0.4414218068122864
v600e	0.413941353559494
mutate	0.379623681306839
kras_protoneoplastic_cell_transformation,...	0.37866097688674927
loss_of_heterozygosity	0.3710901141166687
p.val600glu	0.36764857172966003
nras_protooncogene,_gtpase	0.36736059188842773
kras_protooncogene,_gtpase	0.36403611302375793
p.q61r	0.36346468329429626
mutates	0.3626430928707123
t12;15	0.36259788274765015
v600k	0.3617314398288727
polymerase_chain_reactionsingle_strand	0.3608422577381134
3p21	0.3605020344257355
mutation,_missenses	0.3583070635795593
polymerase_chain_reactionpolymorphism,...	0.35616233944892883
e545k	0.3508622646331787
mutates;	0.350714772939682
mutateal	0.3476957082748413
braf_protooncogene,_serine/threonine_kinase	0.3465832471847534

Table 6.4: Words near the vector of 'c.1799T>A'.

Word	Cosine similarity
braf_protooncogene,_serine/threonine_kinase	0.7897012233734131
kras_protooncogene,_gtpase	0.7098353505134583
v600k	0.6978196501731873
nras_protooncogene,_gtpase	0.6833683848381042
v600	0.626879096031189
t790m	0.6217910051345825
mutates	0.619806170463562
braf_protooncogene,_serine/threonine_kinase...	0.6184836626052856
braf_protoneoplastic_cell_transformation,...	0.6061273217201233
phosphatidylinositol4,5bisphosphate_3kinase...	0.5964162349700928
mutate	0.5953807234764099
braf_protooncogene,_serine/threonine_kinasev600e	0.5880165100097656
mutatepositive	0.5861457586288452
l858r	0.5852635502815247
p.v600e	0.5781044363975525
ret_protooncogene/ptc	0.5680065155029297
vemurafenib	0.5650728344917297
hras	0.5468088388442993
tumor_protein_tumor_protein_p53	0.5445054769515991
g13d	0.5407660007476807

Table 6.5: Words near the vector of 'V600E'.

Word	Number of occurence in PubMed&PMC
c.199-10t	13
c.1393-1g	12
c.1129-5923	12
c.1210-12t	10
c.118-308c	9
c.1534-3c	8
c.1091-2a	8
c.1-25c	8

Table 6.6: Selected examples from the word not in the model because of the minimum count higher than the occurrence of the word.

7 Discussion

The evaluation shows that using the databases from [LLB+17] as evaluation data results in a poor performance. A detailed look in the results shows that the retrieved synonyms contain a large amount of false positive in regards to the evaluation set.

Looking at the Table 6.5 the false positive rate in the top result is very high. On one side, there exist real false positives like 'v600k' (another mutation), genes mentions but also verbs and names appear.

An additional tagger could potentially help sorting out the false positives and reduce the output data.

On the other side, there exists false positives in regards to the evaluation set, that are actually true positives. As in Table 6.5 the name occurring at position 12 in the output 'braf_protooncogene,_serine/threonine_kinasev600e' that originates from 'BRAFV600E' in the text that is not untypical for researches to write (gene and the mutation).

When restricting only on tests where there has been at least one mutation in the result set, the recall is on a good level, while the precision is still not acceptable (see Table 6.3). These results indicate that the top ranked results (being most similar to the input word) are similar, but are not expected mutation mentions. Also without any knowledge in some cases the model placed two words that are just sharing contexts in a similar way and ranked the two words similar in a window of 120 words by co-occurrence compared to other more than a million words.

This example shows that the model can extract strong relationships from the text by containg a large amount of false positives. A fixed threshold could not be used as the cosine similarity has high absolute differences.

This indicates that the two classes "is a mutation" and "is not a mutation" cannot be easily separated based on their current vector representation. When having more vector dimensions eventually there exist one or more dimensions encode for the "is a mutation" relationship.

Additionally the performance is most likely improvable by choosing a smaller context size as well as using more dimensions. A simple logistic regression model based on the vectors and trained by positive and negative "is a mutation mention" relationship has poor performance as well (Area under ROC = 0.58).

As the low frequency words cannot be included, the rare mutation mentions cannot be captured. Also possible key words only occurring with a certain mutation might

occur below the minimum count border. Probably a model with a very low min count can capture the rare mutation mentions.

The overwhelming results achieved in sentence completion tasks and machine translation using word embeddings might come through enough samples in a very well connected everyday language model. The specialized language used in publications with many rare occurrences of individual words probably makes it hard for a model using co-occurence of the context words to learn good vector representations for the words.

8 Outlook and future work

Future key points to improve are:

- Lower min count : include more words, more predictions and also rare mutation mentions are included.
- Smaller window size : the larger the context size the larger the abstraction of the word beeing in focus. A smaller window size might help to separate the different words better.
- Better cleansing : The TERMite and Stanford CoreNLP processing increases the precision and recall, the cleansing of the data can be stronger and normalizing e.g. 'mutates' and 'mutate' and 'mutation' occur on a level were one form should be enough (Table 6.4)
- Add rules for tagging known mutations : As this model is not a tagger, this information can improve one. By querying the model for a vector the similar words can be processed and decided whether one or more synonyms of the term to tag is actually a mutation. Based on that the tagger can then make an informed decision using the word embeddings as additional context information.
- More data: It is shown that most of the mutations are not contained in the used corpus. Using a larger corpus with also the rare occurring mutation mentions often enough, will enable the model to train appropriate vectors.

Danksagung

Ich möchte mich bei der Firma Roche Diagnostics GmbH bedanken, dass ich meine Masterarbeit in Ihrem Unternehmen absolvieren konnte. Ganz besonders möchte ich mich bei meinen Betreuenden Dr. Thomas Schödl, Dr. Stefanie Kaufmann und Martin Baron von der Firma Roche Diagnostics GmbH, sowie bei meiner Betreuerin Prof. Dr. Hannah Bast von der Albert-Ludwigs-Universität Freiburg bedanken.

Für die tolle Zeit in Bayern bedanke ich mich bei dem gesamten Team der Wissenschaftlichen Informationsdienste Penzberg. Vielen Dank an Iván San Antonio Martinez vom Team Integrated Data Warehouse für die Bereitstellung der Rechenleistung und benötgter Pakete.

Außerdem möchte ich mich bei der Firma SciBite Limited für die Bereitstellung der Lizenz für ihre Software TERMite bedanken.

Bibliography

[ABH+12] Alberts, Bruce ; Bray, Dennis ; Hopkin, Karen ; Johnson, Alexander ; Lewis, Julian ; Raff, Martin ; Roberts, Keith ; Walter, Peter: *Lehrbuch der molekularen Zellbiologie*. John Wiley & Sons, 2012

[AMAS18] Akiyama, Michiko ; Matsuda, Yoko ; Arai, Tomio ; Saeki, Hidehisa: Clinicopathological characteristics of malignant melanomas of the skin and gastrointestinal tract. In: *Oncology letters* 16 (2018), Nr. 2, S. 2675–2681

[BDS+08] Bundschus, Markus ; Dejori, Mathaeus ; Stetter, Martin ; Tresp, Volker ; Kriegel, Hans-Peter: Extraction of semantic biomedical relations from text using conditional random fields. In: *BMC Bioinformatics* 9 (2008), 8, Nr. 1, S. 1–14. – ISSN 1471–2105

[BTZ+12] Bollag, Gideon ; Tsai, James ; Zhang, Jiazhong ; Zhang, Chao ; Ibrahim, Prabha ; Nolop, Keith ; Hirth, Peter: Vemurafenib: the first drug approved for BRAF-mutant cancer. In: *Nature Reviews Drug Discovery* 11 (2012), 12, Nr. 11, S. 873–886. – ISSN 1474–1776

[CBR+07] Caporaso, J G. ; Baumgartner, W A. ; Randolph, D A. ; Cohen, K B. ; Hunter, L: MutationFinder: a high-performance system for extracting point mutation mentions from text. In: *Bioinformatics* 23 (2007), 7, Nr. 14, S. 1862–1865. – ISSN 1367–4803

[CBU+] Cejuela, Juan M. ; Bojchevski, Aleksandar ; Uhlig, Carsten ; Bekmukhametov, Rustem ; Karn, Sanjeev K. ; Mahmuti, Shpend ; Baghudana, Ashish ; Dubey, Ankit ; Satagopam, Venkata P. ; Rost, Burkhard: nala: text mining natural language mutation mentions. In: *Bioinformatics* , S. btx083. – ISSN 1460–2059

[CCKP16] Chiu, Billy ; Crichton, Gamal ; Korhonen, Anna ; Pyysalo, Sampo: How to Train good Word Embeddings for Biomedical NLP. (2016), S. 166–174

[CFL13] Clark, A. ; Fox, C. ; Lappin, S.: *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley, 2013 (Blackwell Handbooks in Linguistics). – ISBN 9781118448670

[DS14] Dracopoli, Nicholas C. ; Streicher, Katie: *Genomic Biomarkers for Pharmaceutical Development*. 2014. – 23–49 S. – ISBN 9780123973368

[FFJ+16] Fornander, Linnea ; Friberg, Marc ; Johansson, Vida ; Lindh-Håård, V ; Ohlsson, Pontus ; Palm, Ida: Generating Synonyms Using Word Vectors and an Easy-to-Read Corpus. (2016)

[FPK+10] Flaherty, Keith T. ; Puzanov, Igor ; Kim, Kevin B. ; Ribas, Antoni ; McArthur, Grant A. ; Sosman, Jeffrey A. ; O'Dwyer, Peter J. ; Lee, Richard J. ; Grippo, Joseph F. ; Nolop, Keith ; Chapman, Paul B.: Inhibition of Mutated, Activated BRAF in Metastatic Melanoma. In: *The New England Journal of Medicine* 363 (2010), 10, Nr. 9, S. 809–819. – ISSN 0028–4793

[FRHLL+18] Fernández-Ramírez, Fernando ; Hurtado-López, Luis M. ; López, Mario A. ; Martínez-Peñafiel, Eva ; Herrera-González, Norma E. ; Kameyama, Luis ; Sepúlveda-Robles, Omar: BRAF 1799T> A Mutation Frequency in Mexican Mestizo Patients with Papillary Thyroid Cancer. In: *BioMed research international* 2018 (2018)

[GCWL14] Guo, Jiang ; Che, Wanxiang ; Wang, Haifeng ; Liu, Ting: Revisiting Embedding Features for Simple Semi-supervised Learning. (2014), S. 110–120

[GL14] Goldberg, Yoav ; Levy, Omer: word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. (2014)

[KLKK18] Kim, Sunkyu ; Lee, Heewon ; Kim, Keonwoo ; Kang, Jaewoo: Mut2Vec: distributed representation of cancerous mutations. In: *BMC Medical Genomics* 11 (2018), Nr. Suppl 2, S. 33

[LGD15] Levy, Omer ; Goldberg, Yoav ; Dagan, Ido: Improving distributional similarity with lessons learned from word embeddings. In: *Transactions of the Association for Computational Linguistics* 3 (2015)

[Lim18] Limited, SciBite. *TERMite API*. 2018

[LKC+18] Lee, Kyubum ; Kim, Byounggun ; Choi, Yonghwa ; Kim, Sunkyu ; Shin, Wonho ; Lee, Sunwon ; Park, Sungjoon ; Kim, Seongsoon ; Tan, Aik C. ; Kang, Jaewoo: Deep learning of mutation-gene-drug relations from the literature. In: *BMC Bioinformatics* 19 (2018), Nr. 1, S. 21

[LLB+17] Landrum, Melissa J. ; Lee, Jennifer M. ; Benson, Mark ; Brown, Garth R. ; Chao, Chen ; Chitipiralla, Shanmuga ; Gu, Baoshan ; Hart, Jennifer ; Hoffman, Douglas ; Jang, Wonhee [u. a.]: ClinVar: improving access to variant interpretations and supporting evidence. In: *Nucleic acids research* 46 (2017), Nr. D1, S. D1062–D1067

[LMP01] Lafferty, John ; McCallum, Andrew ; Pereira, Fernando C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001), 1

[Ltd18] Ltd, F. Hoffmann-La R. *What is Personalised Healthcare all about?* 2018

[MA13] Moen, SPFGH ; Ananiadou, Tapio Salakoski2 S.: Distributional semantics resources for biomedical text processing. In: *Proceedings of the 5th International Symposium on Languages in Biology and Medicine, Tokyo, Japan* (2013)

[MAM08] Mansouri, Alireza ; Affendey, Lilly S. ; Mamat, Ali: Named Entity Recognition Approaches. (2008)

[MCCD13] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. (2013)

[Mey16] Meyer, David. *How exactly does word2vec work?* 2016

[MH09] Mnih, Andriy ; Hinton, Geoffrey E.: A scalable hierarchical distributed language model. In: *Advances in neural information processing systems*, 2009, S. 1081–1088

[MIS14] for Mucosal Immunology (SMI), Society. *Single Nucleotide Polymorphism (SNP) Allele Frequency DNA Pools.* 2014

[MMS99] Manning, Christopher D. ; Manning, Christopher D. ; Schütze, Hinrich: *Foundations of statistical natural language processing.* MIT press, 1999

[MSB+14] Manning, Christopher D. ; Surdeanu, Mihai ; Bauer, John ; Finkel, Jenny ; Bethard, Steven J. ; McClosky, David: The Stanford CoreNLP Natural Language Processing Toolkit. In: *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, S. 55–60

[MSC+13] Mikolov, Tomas ; Sutskever, Ilya ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Distributed Representations of Words and Phrases and their Compositionality. (2013)

[MU17] of Medicine (US), National L. *Yearly Citation Totals from 2017 MEDLINE/PubMed Baseline.* 2017

[MUGHR13a] of Medicine (US). Genetics Home Reference, Genes National L. *HBB gene - Sickle cell disease.* 2013

[MUGHR13b] page: National Library of Medicine (US). Genetics Home Reference, Help Me Understand G. *What is a gene?* 2013

[Pal18] of California Museum of Paleontology, University. *Understanding Evolution.* 2018

[PMA+10] Pakhomov, Serguei ; McInnes, Bridget ; Adam, Terrence ; Liu, Ying ; Pedersen, Ted ; Melton, Genevieve B.: Semantic Similarity and Relatedness between Clinical Terms: An Experimental Study. In: *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium* 2010 (2010), 10, S. 572–6

[PSP+06] Plake, Conrad ; Schiemann, Torsten ; Pankalla, Marcus ; Hakenberg, Jörg ; Leser, Ulf: AliBaba: PubMed as a graph. In: *Bioinformatics* 22 (2006), 6, Nr. 19, S. 2444–2445. – ISSN 1367–4803

[RE15] Rodriguez-Esteban, Raul: Biocuration with insufficient resources and fixed timelines. In: *Database* 2015 (2015), S. bav116. – ISSN 1758–0463

[Ron14] Rong, Xin: word2vec Parameter Learning Explained. (2014)

[SBSPM15] Segura-Bedmar, Isabel ; Suárez-Paniagua, Víctor ; Martínez, Paloma: Exploring Word Embedding for Drug Name Recognition. (2015), S. 64–72

[SCO09] Sarmento, Luís ; Carvalho, Paula ; Oliveira, Eugénio: Exploring the vector space model for finding verb synonyms in Portuguese. In: *Proceedings of the International Conference RANLP-2009* (2009), 9

[She18] Shelly. *Sickle Cell Anemia- Types, Symptoms, Causes, Diagnosis and Treatment.* 2018

[SKRC10] Shvachko, Konstantin ; Kuang, Hairong ; Radia, Sanjay ; Chansler, Robert: The hadoop distributed file system. In: *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on* Ieee, 2010, S. 1–10

[SSWT17] Stergiou, Stergios ; Straznickas, Zygimantas ; Wu, Rolina ; Tsioutsiouliklis, Kostas: Distributed Negative Sampling for Word Embeddings. In: *AAAI* (2017)

[SUY+17] Senel, Lutfi K. ; Utlu, Ihsan ; Yucesoy, Veysel ; Koc, Aykut ; Cukur, Tolga: Semantic Structure and Interpretability of Word Embeddings. (2017)

[SWK+01] Sherry, Stephen T. ; Ward, M-H ; Kholodov, M ; Baker, J ; Phan, Lon ; Smigielski, Elizabeth M. ; Sirotkin, Karl: dbSNP: the NCBI database of genetic variation. In: *Nucleic acids research* 29 (2001), Nr. 1, S. 308–311

[TKF+11] Thomas, Philippe E. ; Klinger, Roman ; Furlong, Laura I. ; Hofmann-Apitius, Martin ; Friedrich, Christoph M.: Challenges in the association of human single nucleotide polymorphism mentions

with unique database identifiers. In: *BMC Bioinformatics* 12 (2011), 11, Nr. S4, S. 1–18. – ISSN 1471–2105

[TRH+16] Thomas, Philippe ; Rocktäschel, Tim ; Hakenberg, Jörg ; Lichtblau, Yvonne ; Leser, Ulf: SETH detects and normalizes genetic variants in text. In: *Bioinformatics (Oxford, England)* 32 (2016), Nr. 18, S. 2883–5. – ISSN 1367–4803

[TSV+12] Thomas, Philippe ; Starlinger, Johannes ; Vowinkel, Alexander ; Arzt, Sebastian ; Leser, Ulf: GeneView: a comprehensive semantic search engine for PubMed. In: *Nucleic Acids Research* 40 (2012), 12, Nr. W1, S. W585–W591. – ISSN 0305–1048

[UBP17] Unanue, Inigo J. ; Borzeshi, Ehsan Z. ; Piccardi, Massimo: Recurrent neural networks with specialized word embeddings for health-domain named-entity recognition. (2017)

[VLBW+13] Van Landeghem, Sofie ; Björne, Jari ; Wei, Chih-Hsuan ; Hakala, Kai ; Pyysalo, Sampo ; Ananiadou, Sophia ; Kao, Hung-Yu ; Lu, Zhiyong ; Salakoski, Tapio ; Van de Peer, Yves [u. a.]: Large-Scale Event Extraction from Literature with Multi-Level Gene Normalization. In: *PLoS ONE* 8 (2013), Nr. 4, S. e55814

[Voh16] Vohra, Deepak: Apache Parquet. In: *Practical Hadoop Ecosystem*. Springer, 2016, S. 325–335

[WPF+17] Wei, Chih-Hsuan ; Phan, Lon ; Feltz, Juliana ; Maiti, Rama ; Hefferon, Tim ; Lu, Zhiyong: tmVar 2.0: Integrating genomic variant information from literature with dbSNP and ClinVar for precision medicine. In: *Bioinformatics* (2017). – ISSN 1367–4803

[WST+17] Westergaard, David ; Stærfeldt, Hans-Henrik ; Tønsberg, Christian ; Jensen, Lars J. ; Brunak, Søren: Text mining of 15 million full-text scientific articles. In: *bioRxiv* (2017), S. 162099

[WXJ+15] Wu, Yonghui ; Xu, Jun ; Jiang, Min ; Zhang, Yaoyun ; Xu, Hua: A Study of Neural Word Embeddings for Named Entity Recognition in Clinical Text. In: *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium* 2015 (2015), S. 1326–33

[YV14] Yepes, Antonio J. ; Verspoor, Karin: Mutation extraction tools can be combined for robust recognition of genetic variants in the literature. In: *F1000Research* 3 (2014), S. 18. – ISSN 2046–1402

[ZCF+10] Zaharia, Matei ; Chowdhury, Mosharaf ; Franklin, Michael J. ; Shenker, Scott ; Stoica, Ion: Spark: Cluster computing with working sets. In: *HotCloud* 10 (2010), 10