CLASSIFYING DISTINCT DATA TYPES: TEXTUAL STREAMS PROTEIN SEQUENCES AND GENOMIC VARIANTS

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

DOKTOR DER NATURWISSENSCHAFTEN

Dr. rer. nat.

genehmigte Dissertation
von

M. Sc. Damianos P. Melidis

geboren am 22. September 1988, in Athen, Griechenland

Hannover, Deutschland, 2023

Referent: Prof. Dr. techn. Wolfgang Nejdl
Korreferent: Prof. Dr. rer. nat. Tim Kacprowski
Tag der Promotion: 22 März 2023

Σε όσους μου έδωσαν την εμπιστοσύνη, πως μπορώ να ξεπεράσω τους εδικούς μου "Λαιστρυγόνας" και "Κύκλωπας", στα ταξίδια για τις Ιθάκες της ζώης μου: οικογένεια, δεύτερους γονείς (Λευτέρη και Μαριάνθη), έμπιστους φίλους και στην γλυκιά Τέτη.

ABSTRACT

Artificial Intelligence (AI) is an interdisciplinary field combining different research areas with the end goal to automate processes in the everyday life and industry. The fundamental components of AI models are an "intelligent" model and a functional component defined by the end-application. That is, an intelligent model can be a statistical model that can recognize patterns in data instances to distinguish differences in between these instances. For example, if the AI is applied in car manufacturing, based on an image of a part of a car, the model can categorize if the car part is in the front, middle or rear compartment of the car, as a human brain would do. For the same example application, the statistical model informs a mechanical arm, the functional component, for the current car compartment and the arm in turn assembles this compartment, of the car, based on predefined instructions, likely as a human hand would follow human brain neural signals. A crucial step of AI applications is the *classification* of input instances by the intelligent model.

The classification step in the intelligent model pipeline allows the subsequent steps to act in similar fashion for instances belonging to the same category. We define as classification the module of the intelligent model, which categorizes the input instances based on predefined human-expert or data-driven produced patterns of the instances. Irrespectively of the method to find patterns in data, classification is composed of four distinct steps: (i) input representation, (ii) model building (iii) model prediction and (iv) model assessment. Based on these classification steps, we argue that applying classification on distinct data types holds different challenges.

In this thesis, I focus on challenges for three distinct classification scenarios: (i) *Textual Streams*: how to advance the *model building* step, commonly used for static distribution of data, to classify textual posts with transient data distribution? (ii) *Protein Prediction*: which biologically meaningful information can be used in the *input representation* step to overcome the limited training data challenge? (iii) *Human Variant Pathogenicity Prediction:* how to develop a classification system for functional impact of human variants, by providing standardized and well accepted evidence for the classification outcome and thus enabling the *model assessment* step?

To answer these research questions, I present my contributions in classifying these different types of data: `temporalMNB`: I adapt the sequential prediction with expert advice paradigm to optimally aggregate complementary distributions to enhance a Naive Bayes model to adapt on drifting distribution of the characterisitcs of the textual posts. `dom2vec`: our proposal to learn embedding vectors for the protein domains using self-supervision. Based on the high performance achieved by the `dom2vec` embeddings in quantitative intrinsic assessment on the captured biological information, I provide example evidence for an *analogy* between the local linguistic features in natural languages and the domain structure and function information in domain architectures. Last, I describe `GenOtoScope` bioinformatics software tool to automate standardized evidence-based criteria for pathogenicity impact of variants associated with hearing loss. Finally, to increase the practical use of our last contribution, I develop easy-to-use software interfaces to be used, in research settings, by clinical diagnostics personnel.

Key words *classification, textual streams, concept drifts, feature drifts, ensemble learning, time series, protein domain architectures, word embeddings, quantitative quality assessment, SCOPe secondary structure class, enzymatic commission class, human genomic variants, hearing loss, ACMG/AMP classification, bioinformatics, clinical diagnostics*

ZUSAMMENFASSUNG

Künstliche Intelligenz (KI) ist ein interdisziplinäres Gebiet, das verschiedene Forschungsbereiche mit dem Ziel verbindet, Prozesse im Alltag und in der Industrie zu automatisieren. Die grundlegenden Komponenten von KI-Modellen sind ein "intelligentes" Modell und eine durch die Endanwendung definierte funktionale Komponente. Das heißt, ein intelligentes Modell kann ein statistisches Modell sein, das Muster in Dateninstanzen erkennen kann, um Unterschiede zwischen diesen Instanzen zu unterscheiden. Wird die KI beispielsweise in der Automobilherstellung eingesetzt, kann das Modell auf der Grundlage eines Bildes eines Autoteils kategorisieren, ob sich das Autoteil im vorderen, mittleren oder hinteren Bereich des Autos befindet, wie es ein menschliches Gehirn tun würde. Bei der gleichen Beispielanwendung informiert das statistische Modell einen mechanischen Arm, die funktionale Komponente, über den aktuellen Fahrzeugbereich, und der Arm wiederum baut diesen Bereich des Fahrzeugs auf der Grundlage vordefinierter Anweisungen zusammen, so wie eine menschliche Hand den neuronalen Signalen des menschlichen Gehirns folgen würde. Ein entscheidender Schritt bei KI-Anwendungen ist die *Klassifizierung* von Eingabeinstanzen durch das intelligente Modell.

Unabhängig von der Methode zum Auffinden von Mustern in Daten besteht die Klassifizierung aus vier verschiedenen Schritten: (i) Eingabedarstellung, (ii) Modellbildung, (iii) Modellvorhersage und (iv) Modellbewertung. Ausgehend von diesen Klassifizierungsschritten argumentiere ich, dass die Anwendung der Klassifizierung auf verschiedene Datentypen unterschiedliche Herausforderungen mit sich bringt.

In dieser Arbeit konzentriere ich uns auf die Herausforderungen für drei verschiedene Klassifizierungsszenarien: (i) *Textdatenströme*: Wie kann der Schritt der *Modellerstellung*, der üblicherweise für eine statische Datenverteilung verwendet wird, weiterentwickelt werden, um die Klassifizierung von Textbeiträgen mit einer instationären Datenverteilung zu erlernen? (ii) *Proteinvorhersage*: Welche biologisch sinnvollen Informationen können im Schritt der *Eingabedarstellung* verwendet werden, um die Herausforderung der begrenzten Trainingsdaten zu überwinden? (iii) *Vorhersage der Pathogenität menschlicher Varianten*: Wie kann ein Klassifizierungssystem für die funktionellen Auswirkungen menschlicher Varianten entwickelt werden, indem standardisierte und anerkannte Beweise für das Klassifizierungsergebnis bereitgestellt werden und somit der Schritt der *Modellbewertung* ermöglicht wird?

Um diese Forschungsfragen zu beantworten, stelle ich meine Beiträge zur Klassifizierung dieser verschiedenen Datentypen vor: `temporalMNB`: Verbesserung des Naive-Bayes-Modells zur Klassifizierung driftender Textströme durch Ensemble-Lernen. `dom2vec`: Lernen von Einbettungsvektoren für Proteindomänen durch Selbstüberwachung. Auf der Grundlage der berichteten Ergebnisse liefere ich Beispiele für eine *Analogie* zwischen den lokalen linguistischen Merkmalen in natürlichen Sprachen und den Domänenstruktur- und Funktionsinformationen in Domänenarchitekturen. Schließlich beschreibe ich ein bioinformatisches Softwaretool, `GenOtoScope`, zur Automatisierung standardisierter evidenzbasierter Kriterien für die Pathogenitätsauswirkungen von Varianten, die mit angeborener Schwerhörigkeit in Verbindung stehen.

Schlagworte *Klassifizierung, Textströme, Konzeptdrift, Featuresdrift, Ensemble-Lernen, Zeitserien, Proteindomänenarchitekturen, Word Embeddings, quantitative Qualitätsbewertung, SCOPe-Sekundärstrukturklasse, Enzymkommissionsklasse, humangenomische Varianten, Hörverlust, ACMG/AMP-Klassifizierung, Bioinformatik, klinische Diagnostik*

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor Prof. Wolfgang Nejdl for giving me the opportunity to pursue the PhD opportunity. By his supervision I have learned how to be a self-sustained researcher, who works as much as he can. Equally, I would like to thank Prof. Tim Kacprowski for his time to review my thesis and be the second examiner of my dissertation committee. Then, I would like to thank the collaborators of the research projects that I worked for. For BacData, I would like to thank Sarah Pohl and Matthias Steglich for their fruitful collaboration. For the Cochlear Implant (CI) project, I am grateful to work with the Functional Genomics group of the Human Genetics Department and specifically: Anja Schöner-Heinisch, Christian Landgraf, Gunnar Schimdt and Bernd Auber; without their patient collaboration I could not have one my most practical contribution of this thesis. I would also like to thank Prof. Stefano Ceri for giving me the opportunity to work with his team, and especially with Pietro Pinoli, during the first period of the pandemic.

Second, I would like to thank my family for the great support and patience through all these years being abroad from my homeland. I would like to thank my father, Panayiotis, for his creativity and intellectualism, my mothers, Maria and Marianthi, for their love and positive thinking and Lefteris for being always happy to take care of me. My sister, Anastasia, for always respecting and loving me and my two uncles, Yianni and Michali, for always loving me and be positive for my dreams. I would also deeply like to thank my long-lasting friends, Christos, Christophoros, Giorgos, Giorgos, Panayiotis, Giannis, Manolis, Adreas, Vangelis, Odysseas, Michalis, Carolos, Eirini, Elina, Giorgos, Liren, Omar, Nicole, Christos, Sotiris, Nicol and my girlfriend, Tetie, for accepting me, as I truly am and for making me to think positive during this PhD journey! I would also love to thank Giorgos and Prof. Christos Makris for their support during my master and doctorate studies, their trust on my skills really meant a lot. Furthermore, I am honestly grateful to Vangelis for this trust on my intellectual skills during my young ages, his confidence on me moved me forward.

Last but not least, I would like to thank my L3S colleagues for creating a friendly research environment. Especially, I would like to thank Asme, Alvaro, Jan, Miro, Simon, Michelle, Ngan, Huyen, Tai, Jurek, Lijun, Amir, Sajjad, Nicolas, Markus, Monty, for their co-supervision, exchange of ideas/views and time to go outside the office and have fun! Finally, I would like to thank Michaela and Angelika for always having time to listen to me and appreciate my true personality.

FOREWORD

To compose this thesis, I have used, word by word, the published manuscripts of the three most important research outputs of my doctorate work.

The research contributions, presented in this thesis, have been published at various conferences and journals, as follows.

Chapter 3 presents our work, `temporalMNB`, on learning algorithms for textual streams over concept drift events:

- Melidis, D. P., Spiliopoulou, M., & Ntoutsi, E. (2018, October). Learning under feature drifts in textual streams. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (pp. 527-536) (Full paper - research track)

  DOI: https://doi.org/10.1145/3269206.3271717. (Melidis et al., 2018b)

Chapter 4 presents our work, `dom2vec`, on learning and evaluating neural representations for protein domains:

- Melidis, D. P., & Nejdl, W. (2021). Capturing Protein Domain Structure and Function Using Self-Supervision on Domain Architectures. Algorithms, 14(1), 28 (Full journal paper)

  DOI: https://doi.org/10.3390/a14010028. (Melidis and Nejdl, 2021)

I describe my method, GenOtoScope, to automate ACMG/AMP criteria for pathogenicity classification of variants associated with congenital hearing loss in Chapter 5:

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Bernd Auber, Wolfgang Nejdl (2021, August). GenOtoScope: Automated annotation of variants associated with hereditary hearing loss. European Symposium of Human Genetics (Poster presentation). (*) First-author equal contribution. (Landgraf et al., 2021)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Alisa Förster, Anke Lesinski-Schiedat, Bernd Auber, Wolfgang Nejdl (2021, August). GenOtoScope: Automated annotation of variants associated with hereditary hearing

loss. European Symposium of Human Genetics (Poster presentation). (*) First-author equal contribution. (Landgraf et al., 2022a)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Alisa Förster, Anna-Lena Katzke, Anke Lesinski-Schiedat, Bernd Auber ‡, Wolfgang Nejdl ‡. GenOtoScope: Towards automating ACMG classification of variants associated with congenital hearing loss. (*) First-author equal contribution. (Landgraf et al., 2022b)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Anke Lesinski-Schiedat, Bernd Auber ‡, Wolfgang Nejdl ‡. GenOtoScope: Towards automating ACMG classification of variants associated with congenital hearing loss. (*) First-author equal contribution. (‡) Last-author equal contribution. PLoS Computational Biology (Full journal paper) DOI: https://doi.org/10.1371/journal.pcbi.1009785. (Melidis et al., 2022)

The published work (Melidis et al., 2022) that was used, word by word, to create Chapter 5, is a shared first-authorship between Dr. med. Christian Landgraf and myself (Damianos Melidis M.Sc.). That is, the contributions of Dr. med. Landgraf were in the data curation, method investigation, project administration, supervision of M.Sc. Melidis. Besides, Dr. med. Landgrad contributed to the validation of variant classification based on manual investigation and in reviewing and editing the manuscript. M.Sc. Melidis contributed to the `GenOtoScope` publication in the formal analysis and investigation of the methodology, in the software development and validation of the methodology. Also M.Sc. Melidis supplied software solutions to visualize all results of the variant classification and he committed to design all flow-chart workflows, and in general images and tables, of the publication. Finally, M.Sc. Melidis wrote the original draft of the manuscript, which then he reviewed and edited the manuscript based on all co-authors and reviewers comments to create the very final form of the manuscript.

The complete list of publications during my PhD follows:

Journal articles

- Melidis, D. P., & Nejdl, W. (2021). Capturing Protein Domain Structure and Function Using Self-Supervision on Domain Architectures. Algorithms, 14(1), 28 (Full journal paper)

  DOI: https://doi.org/10.3390/a14010028. (Melidis and Nejdl, 2021)

- Canakoglu, A., Pinoli, P., Bernasconi, A., Alfonsi, T., Melidis, D. P., & Ceri, S. (2021). ViruSurf: an integrated database to investigate viral

sequences. Nucleic acids research, 49(D1), D817-D824. (Full journal paper)

DOI: https://doi.org/10.1093/nar/gkaa846. (Canakoglu et al., 2021)

Conference publications

- Melidis, D. P., Spiliopoulou, M., & Ntoutsi, E. (2018, October). Learning under feature drifts in textual streams. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (pp. 527-536). (Full paper - research track)

  DOI: https://doi.org/10.1145/3269206.3271717. (Melidis et al., 2018b)

- Melidis, D. P., Campero, A. V., Iosifidis, V., Ntoutsi, E., & Spiliopoulou, M. (2018, June). Enriching lexicons with ephemeral words for sentiment analysis in social streams. In Proceedings of the 8th international conference on web intelligence, mining and semantics (pp. 1-8). (Full paper)

  DOI: https://doi.org/10.1145/3227609.3227664. (Melidis et al., 2018a)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Bernd Auber, Wolfgang Nejdl (2021, August). GenOtoScope: Automated annotation of variants associated with hereditary hearing loss. European Symposium of Human Genetics (Poster presentation). (*) First-author equal contribution. (Landgraf et al., 2021)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Alisa Förster, Anke Lesinski-Schiedat, Bernd Auber, Wolfgang Nejdl (2021, August). GenOtoScope: Automated annotation of variants associated with hereditary hearing loss. European Symposium of Human Genetics (Poster presentation). (*) First-author equal contribution. (Landgraf et al., 2022a)

- Christian Landgraf *, Damianos P. Melidis *, Gunnar Schmidt, Anja Schöner-Heinisch, Sandra von Hardenberg, Alisa Förster, Anna-Lena Katzke, Anke Lesinski-Schiedat, Bernd Auber ‡, Wolfgang Nejdl ‡. GenOtoScope: Towards automating ACMG classification of variants associated with congenital hearing loss. (*) First-author equal contribution. (Landgraf et al., 2022b)

Under submission

- Szymon P. Szafranski, Damianos P. Melidis, Lisan D. Püttmann, Wolfgang Nejdl and Meike Stiesch. Profiling of conserved protein domains revealed phylogenetic and functional diversity of oral bacteriophages. (Before submission stage)

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Artificial intelligence (AI) is an interdisciplinary field that combines different research areas from the natural and technological sciences, to automate processes in everyday life and industry. The basic components of an AI model are two: (i) a statistical model that learns what is the correct output for the given input based on *training examples*, imitating a human being, (ii) based on the application, a functional component which will act mechanically or electronically to produce a practical result, depending on the output of the first component. In 21st century, example applications of AI are: the email spam detection, the self-driving cars, the self-regulated pipeline steps in industry and the three-dimensional structure prediction of a protein. The automated *classification* of input is a mandatory step of the first component for these applications, as it enables the AI model to act in a similar fashion for all inputs belonging to the same class.

Automated classification systems were first build using human-crafted knowledge in the form of if-then clauses to categorize input instances based on their characteristics. These *rule-based* approaches were first used in *expert systems* from 1970's Leondes (2001). Through the following decades and due to the success of the expert systems researchers aim to classify data from scientific with no *a-priori* expert knowledge (astrophysics, molecular biology, and medicine). In order to fulfill this need a new research field was born named *Machine Learning* (ML).

In general, machine learning field seeks to develop methods to extract useful patterns from input data, in an automatic fashion, using mathematical models Bishop (2006). Two common paradigms in machine learning field is the *supervised* and *unsupervised* learning. In the first, the model requires the input data to be labeled with a discriminating characteristic, *target class*. Subsequently, the model aims to discover, *learn*, patterns to relate the input characteristics to the target class with the end-goal,

the mapping achieving the best *prediction* performance. For the latter paradigm, the model does not demand an input labeled with a discriminating characteristic, next the model learns hidden motives to separate the input data into distinct subgroups with the minimum error.

Next, I summarize the common statistical models used for classification. Classification models can be distinguished in *eager* and *lazy* learners, by their order in applying the model building and prediction. Given the input data, eager learners build their prediction model *before* the arrival of a new data instance to be classified, whereas, lazy learners create their model *at the point of* the arrival of the new data instance. Based on Friedman (2017), classical example of eager learners are Bayesian models such (*Naive Bayes* Hand and Yu (2001) or Bayes network classifiers Friedman et al. (1997)), entropy-based models (decision trees Hunt et al. (1966); Praagman (1985); Quinlan (1986)), support vector machines (SVM) Boser et al. (1992) and artificial neural networks (ANN) Rosenblatt (1957). Briefly, an ANN is a model which combines the input characteristics to predict the output using threshold functions, like a fully connected layer of synapses in the human brain. An example of lazy learners is the k-nearest neighbor (k-NN) classifier. Following the intuition that combining more experts the prediction performance improves, *ensemble learning* deploys a set of the previous single learners and acquires their prediction for a given input instance, then it averages their predictions in the final classification outcome.

These classical examples of ML have learning limitations. That is, these classical models can learn the hidden relation between input and output if the underlying relation is *linear*. In a nutshell, we say that there is a linear relation between input and output, when there exists a linear combination of input instance characteristics that equals the output. However, most systems in nature are *non-linear*. For example, the prediction of the disease state of a patient based on his health records, as a small change on the patient's health record does not equal a proportional change on her disease state. Machine learning overcomes this challenge by start to developing non-linear methods, for example, SVM with Gaussian kernel function or ANNs with a limited complexity capacity. Such ANNs were developed successfully for the classification of handwritten digits by LeCun et al. (1998). However, these networks needed a large amount of computing power and labeled data set to outperform classic algorithms (such as SVMs). Since 2010, the computing power advances and the availability of large labeled data sets enabled researchers to resume their research and application of ANNs. Resulting to the birth of a new field, deep learning (DL). In deep learning, the neural networks are ANN with multiple number of layers and are organized with distinct architectures. These deep neural networks (DNN) have the property to increase their ability to learn input-output relation as the input data increase exponentially, in contrast to other non-linear methods. DNN have achieved to outperform even the human curators for specific tasks such as: hand-written words recognition Graves and Schmidhuber (2008) or image classification Krizhevsky et al. (2012).

We can separate the classification into four distinct components. The first component, *input representation*, creates a dimensional representation of input data based on their fixed number of characteristics, *features*. The second component, *model building*, is the deployment of a statistical or rules-based model to learn from the input data set, *training set*, which contains the features and the target class. Next, the model will be able to predict the class of unseen instances, *test set*, in the *model prediction* step. This component will output: the predicted class for the set of test instances and any available information on the human-comprehensible reasons for the respective predictions. The last part of the classification is the *model assessment*, at which a domain expert, will validate the predictions, before the actual use in an AI system.

Model assessment step is an optional step in classification process. However, it is an important step, because it can enhance the trust of the end-users by providing human-comprehensible information on the reasons of test prediction during the model prediction step. This information would be significant if the classification model is used in AI application where a human being, *domain expert*, will validate of the predictions before the actual public use. For example, in email spam detection, the input component will accept the features for each email, the title, the text corpus and the target class, "spam" or "no spam" and represent them in multi-dimensional space. Then, the applied model will use a selected *algorithm* to learn the association between the email features and the target class. Finally, the spam expert personnel may review the predictions for a test set of emails to ensure the correctness of the used email features by the model and to validate its performance levels.

Another part of machine learning research, with equal importance to model design, are the properties of the input data. Input data can be grouped by their change over time in: *stationary* and *transient*. Data of the first category are sampled from a stationary distribution, for example when a person enters images of the 0-9 digits as input data into a computer. However, nowadays many of the input data are collections of the same data type over time, known as "data streams". For instance, social media posts about environment or plant consumption measurements of a energy network over time.

Based on the input data characteristics, the availability of labeled data and the need for model assessment by experts, there occur many challenges on applying the standard components of classification. First, designing the model building component for data streams requires that: (i) it can predict at anytime by using limited memory and computing resources (ii) it captures changes on relation between input data and target label with performance guarantees (iii) it can forget outdated data. For molecular biology applications of ML, researchers focus on the feature engineering component because the training data are small in size. Therefore, new methods for the feature engineering component are needed in order to represent the input instances *closer* to already labeled instances. In the specific domain of protein prediction, such new representations should be (i) assessible in a quantitative

manner to allow the domain experts to choose the best representation for the problem at hand (ii) increase the prediction performance and (iii) boost performance even with limited number of labeled input data. Finally, in ML application in medical domain, the absolute need for the model assessment component mandates for simple model building component, such as a rules-based system, that applies expert-based rules. Even if such systems may not be complex statistical models like deep learning models, they can still be challenging. The two challenges are to: (i) create an automatic and efficient implementation of these expert-based rules and (ii) develop an easy-to-use and handful end-user classification application.

In Figure 1.1 we depict the steps for an example classification, the prediction of relevant e-mails. Assuming the user is searching to acquire a means of transport (bike or car) and the shown training and test instances, the classification system follows four steps. In the first, the training instances are represented using vectors, a suitable input form for the learning model. Next, the model is built based on statistics on the training e-mails (doc_1, doc_2 and doc_3). This allows the model to *separate* the relevant from the irrelevant e-mails, illustrated as green and red sub-spaces respectively. In the third step, the model is asked to predict the relevance or not of two testing e-mails (doc_4 and doc_5), the model predicts for both that there are relevant. However, in the last step, the human domain expert will review the predictions and she will investigate the reasons for the prediction of each test email. In our example, the model misclassifies the first testing e-mail (doc_4) as relevant, still the email's subject is not an advertisement on a means of transport. Consequently, even in this simple everyday life classification system, we show that human experts should be included as the final evaluators of prediction before those are available for public use.

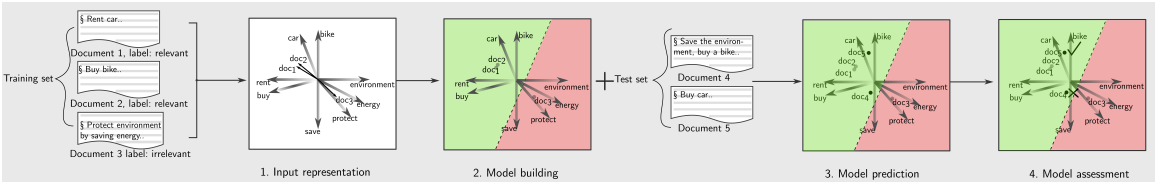


Figure 1.1. Example of classification steps for automatic prediction of relevant e-mails, given that the user is interested in acquiring a new means of transport.

1.2 Research Questions

In this thesis, I focus on distinct challenges in the aforementioned classification components.

Specifically, I research on how to develop: (i) a novel classifier for data stream input which deals with the changes of input data in a data stream (model building), (ii) an alternative representation of input protein data to tackle their low-abundant training sets (input representation), (iii) a rules-based approach to classify human

variants for a specific disease in diagnostic-grade level (model assessment).

In the following sub-chapters, I explain our research questions for each distinct type of input data.

1.2.1 Model Building: Textual Streams

Streams of posts from social media may experience changes on which topics are positive or negative as perceived by the public, equivalently the social media user may change her opinion on words polarity to convey her intended meaning. Therefore, a model to classify the polarity class of a tweet post in a textual stream scenario, imposes a model building component which can capture the different temporal relation of the words with the polarity class and the drifting aspect of the popularity of the topics. These challenges leads us to our first research question:

RQ1 *How to adapt a machine learning model for classification of data streams that exhibit feature-drifts? Given that each feature can be best described by a specific temporal process, how to ensure that this adaptation contain statistical guarantees compared to the feature best-describing process?*

To answer this research question, I focus on an established machine learning model that uses limited memory and computing power in order to be able to predict the polarity of a textual post in anytime fashion. I also constrain on this simple ML model to be able to change its estimated relation of input with the output as the stream experiences temporal changes.

1.2.2 Input Representation: Protein Prediction

Our second research interest is to develop feature engineering methods to lower the data variability in order to achieve well performing models when the training data are limited. This is a common scenario for molecular biology and specifically protein prediction tasks. For this specific field, I aim to develop techniques that transform the input proteins into representations that can be assessed using the metadata of fundamental characteristics of proteins. Besides, I need to ensure that that resulting prediction performance is superior or comparable to the one produced using other established protein representations. I summarize this research question as follows:

RQ2 *For which conceptual level of a protein, we should learn a representation, in order to be quantitatively evaluated? Can such representation boost the prediction performance, on supervised learning tasks, compared to other protein embedding baselines?*

For this research question, we ask for a representation that can be evaluated not only qualitatively but also quantitatively. Therefore, the end-user will be able to validate that the representation captured substantial information on the distinctive biological characteristics of the chosen input representation. This characteristic of our

representation can thus increase the trust of the end-user, before she actually applies *dom2vec* to represent protein instances as input to prediction models. Furthermore, as the assessment of the acquired information is quantitative, the end-user can select the representation instance with the highest information content related to the final prediction task at hand.

1.2.3 Model Assessment: Human Variant Pathogenicity Prediction

More than building a dynamic model or an informative input representation, there is the need to enable the model assessment component through the construction of a human explainable classifier. To do so, how to implement all expert-based criteria for a rules-based system to classify genomic variants for a specific disease? How to enable the practical and convenient usage of such system by the human diagnostic personnel? These criteria compose the following research question:

RQ3 *Given that hereditary hearing loss is a polygenic disease and a commonly used subset of variants a patient contains a large number of variants (in the multiple of 10,000); can we classify genomic variants of patient automatically and in diagnostic-grade manner? How we can provide an practical end-user application for diagnostics personnel?*

Given that a heterogeneous disease may be caused by more than one malfunctioning genes and that the variants found in the range of known genes are approximately 60,000 for one patient, RQ3 asks for an answer with two characteristics. First, I aim to develop a new method on automating domain expert rules in a time and space efficient manner. Second, I draw also our attention on how to make this method practically usable by the diagnostics personnel. The second characteristic is important, as an easy-to-use application for variant classification may have practical benefits in the research diagnostics field.

1.3 Contributions

Table 1.1 presents the conceptual summary of the contributions of the thesis for the three classification scenarios imposed by the research questions: (i) classifying transient textual data streams, (ii) classifying low-abundant labeled protein data sets, (iii) classifying genomic variants associated with hearing loss by a resource-efficient and intuitive application for the clinicians. In the following, I describe in brief these contributions:

Data Type	**Challenge**	**Classification Step**	**Contribution**
Textual Streams	Concept Drift	Model Building	`temporalMNB`
Protein Sequences	Limited Labeled Sets	Input Representation	`dom2vec`
Human Genomic Variants	Assessment of Classification	Model Assessment	GenOtoScope

Table 1.1. Thesis contributions in key-words.

1.3.1 Model Building: Textual Streams

To answer RQ1, I concentrate on textual streams that exhibit temporal changes in the popular topics and the polarity of their textual features (word). On this assumption I created the following contribution:

TemporalMNB

In Chapter 3, I present *TemporalMNB*, an extension of the Multi-nominal Naive Bayes (MNB) to capture the transient nature of textual streams. Our methodology contains two essential steps. First, I create an ensemble of time series experts, each with a complimentary periodicity, to predict the class priors and the conditional probabilities of input text features given the class. In the second step, I encapsulate this ensemble learning in the framework of sequential prediction with experts to acquire guarantees for the prediction performance.

I evaluate our method in real-world data by comparing it to established temporal versions of MNB for three data sets: (i) sentiment classification in Twitter stream and (ii-iii) email category classification for two smaller data sets with known time-drifting features.

1.3.2 Input Representation: Protein Prediction

I have developed *dom2vec* representation of protein domains and applied to protein prediction tasks to answer the *RQ2*.

dom2vec

Based on the analog that maps amino acids to letters and protein domains to words in protein and English (spoken) languages, I describe our method, *dom2vec*, to compute a new representation for protein domains in Chapter 4. To do so, I apply an established word embeddings method to protein domain architectures from InterPro database. As InterPro contains metadata for the most important characteristics of a protein domain, our representation can be evaluated quantitatively using such information. Therefore, I use k nearest neighbor (k-NN) classifier to predict biologically-important aspects of domain such as, secondary structure, enzymatic and GO molecular function, for a given domain based on its representation, embedding vector. To fulfill the second requirement of the RQ2, I input protein domain embeddings in simple neural networks architectures and compare their prediction per-

formance with protein sequence embeddings for three tasks: (i) toxicity, (ii) enzymatic function and (iii) cellular location.

1.3.3 Model Assessment: Human Variant Pathogenicity Prediction

To answer our last research question, *RQ3*, I develop the GenOtoScope bioinformatics tool as follows:

GenOtoScope

I have designed and implemented GenOtoScope tool that accepts as input the variants of one or more patients, it assesses, in automatic fashion, 12 out of the 24 expert-based rules specified for hearing loss. Then it combines the triggered rule to predict the variant pathogenicity class. Finally, the posterior probability of pathogenicity is calculated and the comments for each examined rule are presented in the output file.

I evaluate our method compared to two other established methods for pathogenicity prediction on two data sets: (i) the set of manually annotated variants by the expert panel for hearing loss (ii) the manually annotated variants by the human genetics department of the Hannover medical school (MHH). For both data sets, GenOtoScope outperform the other methods in terms of accuracy and precision. Finally, I interrogate this performance discrepancy computing the ratio of the activation frequency of an expert-based rule by a classification method compared to the manual annotation. By this analysis, I conclude that GenOtoScope triggers the rules in a highly-similar fashion as the manual curation; which is not the case for the other two competing methods.

1.4 Thesis Outline

This thesis is organized as follows: In Chapter 2 I introduce the essential elements needed for a reader to understand the thesis contribution. More specifically, I describe the fundamentals for ML (Classification, Regression, Time Series, Prediction of Expert Advice and Word Embeddings) in section 2. Then, I give a brief introduction and the main challenges for the three data types, targeted in this thesis (Textual Streams, Protein Sequences and Genomic Variants), at section 2.3.4.

The following three chapters (Chapter 3-5) fully describe the thesis three main contributions. That is, Chapter 3 presents `temporalMNB`online classifier for textual streams, Chapter 4 introduces `dom2vec`protein domain embeddings and Chapter 5 describes `GenOtoScope` variant classification software tool. Finally, Chapter 6 summarizes the presented works and outlines future directions based on these works.

To promote open and reproducible science, all presented research works are found at my GitHub repository: Damianos P. Melidis GitHub account.

2 Background

In this chapter, we give a brief introduction on the technical background needed to comprehend the contributions of this thesis. We first introduce the machine learning field focusing on the two main clades of supervised learning, the classification and the regression. This introduction is based on the book "Understanding Machine Learning - From Theory to Algorithms" by Shalev-Shwartz and Ben-David (2014). Then, we present a short intuition on ensemble learning and sequential prediction with expert advice. Next, we summarize common obstacles in learning for transient data. Last, we discuss our main data types: (i) textual streams, (ii) proteins and (iii) genomic variants. For each data type, we give a short introduction of the nature of the data, with the goal to motivate the learning challenges, and a survey of techniques proposed by others to solve our posed research questions.

2.1 Machine Learning

2.1.1 Supervised Learning

Let $\mathcal{X}$ an arbitrary set of data instances and Y to be the set of possible target labels. Then, let the set of pairs, $\mathcal{X} \times \mathcal{Y}$, each containing a data instance and its corresponding label, be the *training set* $S = ((x_1, y_1) \dots (x_m, y_m))$. The output of the learner is expected in the form of a prediction rule, $h : \mathcal{X} \to \mathcal{Y}$. This mapping function is also known as a *predictor*, a *hypothesis* or a *classifier*. This predictor can subsequently be used to assign the value of the target label for an unseen set of unlabeled instances, *test set*, $T = ((x_1 \dots x_k))$. The training and test instances are assumed to be generated by the *same* probability distribution, denoted as $\mathcal{D}$. Let each data point, x_i, that D produces to lie in the d-dimensional space of input characteristics, *feature space*, $x_i \in \mathbb{R}^d$. Concerning the "correct" label generation, we assume that there exists a labeling function $f : \mathcal{X} \to \mathcal{Y}$. It is fundamental for

the learning process to suppose that the learner does hold *no* information about the generating distribution $\mathcal{D}$. Last, we introduce the measure of success of a learner using the quantity known as error of the classifier. The error of h is the probability to draw an random instance, using the distribution $\mathcal{D}$, such as the $h(x)$ is not equal to $f(x)$.

Based on the range of values of $\mathcal{Y}$, supervised learning techniques are subdivided into regression and classification methods. In regression methods, $\mathcal{Y}$ is allowed to be any real number $\mathcal{Y} \in \mathbb{R}$, whereas, $\mathcal{Y} \in \mathbb{N}^+$ for classification methods. For common types of classification are: (i) the binary classification, $\mathcal{Y} \in \{0, 1\}$, and (ii) the multi-class classification where $|\mathcal{Y}| > 2$.

2.1.2 Regression

Linear Regression

Linear regression is an established statistical method to infer the relationship between the features of a data instance and a real valued outcome. Let the domain set to be $\mathcal{X} \in \mathbb{R}^d$, for some d, and the label set be the set of real numbers. We aim to learn a linear hypothesis function $h : \mathbb{R}^d \to \mathbb{R}$ that best captures the "connection" between the features and the target outcome. The hypothesis class of linear regression predictors is set of linear functions to linearly relate the feature value to predict the outcome, as follows:

$$H_{reg} = L_d = \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}, \tag{2.1}$$

where $\boldsymbol{w}$ is the weight vector lying in $\mathbb{R}^d$, $b \in R$ is the bias term and the $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ is the inner product of the features of a instance and the regressor's estimated weight vector. To measure the performance of a regression we need to define a quantity, *loss function*, that penalizes predicted outcome ($h(\boldsymbol{x})$) which is far from the real outcome (y). A common loss function is the squared-loss function, which is defined as:

$$l(h, (\boldsymbol{x}, y)) = (h(\boldsymbol{x}) - y)^2. \tag{2.2}$$

Equivalently, the empirical risk function for the square loss is known as the Mean Square Error and it equals to:

$$L_s(h) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}_i - y_i))^2. \tag{2.3}$$

We will use the "least squares" method to optimize for the Mean Square Error (Equation 2.3). Given a training set S and applying a linear regression hypothesis L_d (2.1), we seek to optimize the:

$$\underset{\boldsymbol{w}}{argmin} L_S(h_{\boldsymbol{w}}) = \underset{\boldsymbol{w}}{argmin} \frac{1}{m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - y_i)^2. \tag{2.4}$$

Using eigenvalue decomposition method, and assuming $m > d$, we can derive the following closed form solution:

$$\hat{w} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}y, \tag{2.5}$$

where $\boldsymbol{X}$ is the matrix created by adding each training instance as a column ($X \in \mathbb{R}^{d\times m}$).

2.1.3 Classification

Naive Bayes

In machine learning theory, we assume that we have no information on the data distribution, $\mathcal{D}$, consequently we are building a model to be able to discriminate/predict the input. Nevertheless, there are cases that it is easier to assume a distribution and then learn its parameters, this paradigm of models in supervised learning is known as *generative models*.

Naive Bayes is the simplest representative of generative models. Assume the vector of a testing instance $\boldsymbol{x} = (x_1, \ldots, x_d)$, where each feature is binary $x_i \in \{0, 1\}$, and we aim to predict its label binary label $y \in \{0, 1\}$. The set of hypotheses, which use the probabilistic approach to find a predictor with minimum error, are known as Bayes optimal predictors. For such classifiers the task is solved by the following hypothesis:

$$h_{Bayes}(\boldsymbol{x}) = \underset{\boldsymbol{y\in\{0,1\}}}{argmax}\mathcal{P}[Y = y|X = \boldsymbol{x}]. \tag{2.6}$$

The probability function $\mathcal{P}[Y = y|X = \boldsymbol{x}]$ needs 2^d parameters to find when $\mathcal{P}[Y = 1|X = \boldsymbol{x}]$ for a specific value of $\boldsymbol{x} \in \{0, 1\}^d$. Therefore, to describe this probability function we need exponential number of examples with respect to the number of features.

Naive Bayes model solves this drawback by the generative assumption, that the features are not related to each other, given the label. This results to:

$$\mathcal{P}[X = \boldsymbol{x}|Y = y] = \prod_{i=1}^{d} \mathcal{P}[X_i = x_i|Y = y]. \tag{2.7}$$

Using this assumption and according to the Bayesian Decision Theory Duda et al. (1973), we have for the Bayes optimal classifier:

$$\begin{aligned} h_{Bayes}(\boldsymbol{x}) &= \underset{\boldsymbol{y\in\{0,1\}}}{argmax}\mathcal{P}[Y = y|X = \boldsymbol{x}] \\ &= \underset{\boldsymbol{y\in\{0,1\}}}{argmax}\mathcal{P}[Y = y]\mathcal{P}[X = \boldsymbol{x}|Y = y] \\ &= \underset{\boldsymbol{y\in\{0,1\}}}{argmax}\mathcal{P}[Y = y]\prod_{i=1}^{d}\mathcal{P}[X_i = x_i|Y = y]. \end{aligned}$$

Therefore, Naive Bayes model needs $2d + 1$ parameters, 2d for the distribution of each feature given each of the two class values and one more parameter for class distribution. This decrease makes the number of parameters linear to the number of features with the drawback of this strong assumption of independence.

Based on Maximum Likelihood principle (MLP), we find the two keys distributions using the observed m training instances $((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$ where $x_i \in \mathbb{R}^d)$ and thus derive the equation of the Naive Bayes predictor:

$$h_{Bayes}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\{\mathbf{0,1}\}}{argmax} \amalg(y) \prod_{i=1}^{d} \amalg_j(x_i|y), \tag{2.8}$$

where

$$q(y) = \frac{\sum_{i=1}^{m}[[y^i = y]]}{m} = \frac{count(y)}{m}, \tag{2.9}$$

with $[[y^i = y]]$ to equal 1 when y^i equals y and 0 otherwise. Thus, the count(y) quantity is the number of times the y label is found in the training instances. Then the maximum likelihood estimate of $q_j(x|y)$, the distribution of the j-th feature given a class y, equals to:

$$q_j(x|y) = \frac{\sum_{i=1}^{m}[[y^{(i)} = y \text{ and } x^{(i)} = x]]}{\sum_{i=1}^{m}[[y^i = y]]} = \frac{count_j(x|y)}{count(y)}, \tag{2.10}$$

where $count_j(x|y) = \sum_{i=1}^{m}[[y_i = y \text{ and } x_i^j = x]]$.

Here, we have presented the Bayes optimal classifier for a Naive Bayes model that aims to predict a binary class $(y \in 0, 1)$. If $\mathcal{Y} \in \mathbb{N}$, then the respective model is known as Multinomial Naive Bayes (MNB) based on the respective multinomial distribution on $\mathcal{Y}$.

k- Nearest Neighbor

The k-Nearest Neighbor algorithm (k-NN) is the most common example of the set of "lazy" learners. Given a training set, k-NN intuition is to use the majority of labels of the closest training instances to the new (testing) instance, in order to predict the class of the test instance. Last k-NN does not construct a hypothesis class, $\mathcal{H}$, to perform the predictions in contrary with the supervised learning models explained previously. This property makes k-NN a lazy learner.

Let the instance domain X equipped with a metric function ρ, $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. This ρ is a function that computes the distance between any pair of points of the domain $\mathcal{X}$. For example, the ρ can be the Euclidean distance in the $\mathcal{X} = \mathbb{R}^d$, that is $\rho(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\| = \sqrt{\sum_{i=1}^{d}(x_i - x_i')^2}$.

The k-NN algorithm with input a sequence of training instances $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$ and a testing instance $\boldsymbol{x} \in \mathbb{R}^d$ will perform the following two steps to predict the testing class. First, it orders the training instances based on their distance to the testing one. This step creates a reordering of the training instances through the $\pi()$ function as: $\pi_1(\boldsymbol{x}), \ldots, \pi_m(\boldsymbol{x})$ so that holds $\rho(\boldsymbol{x}, \boldsymbol{x}_{\pi_i(\boldsymbol{x})}) \leq \rho(\boldsymbol{x}, \boldsymbol{x}_{\pi_{i+1}(\boldsymbol{x})})$, for all $i < m$. In the second step, k-NN will predict as class for the testing $\boldsymbol{x}$ the majority label among the k nearest neighbor of $\boldsymbol{x}$, $\{y_{\pi_i(\boldsymbol{x})} : i \leq k\}$.

Even if, k-NN is efficient to work for large data sets, there are many cases that not all of the features are informative or that the distance of the testing to the training instance holds not all the information needed to predict $\boldsymbol{x}$ class. Therefore, the k-NN is used as a baseline classifier for performance comparison to another advanced learning technique. Also it is used as a benchmark classifier to examine the homogeneity of the labels with respect to the domain dimensions (features).

2.1.4 Ensemble Learning

Ensemble learning is the sub-field of machine learning that investigates how to combine distinct learning models, *experts*/ weak classifiers, to improve the performance of the aggregate predictor composed by these sub-models, also known as the *ensemble of classifiers* Dietterich (2000). The main intuition is that each expert can learn the search space in *its own way* to find the best hypothesis function, therefore by combining (most commonly by voting) such hypothesis function produced by each expert, the ensemble can improve its prediction performance compared to a single expert. Interestingly, Hansen and Salamon (1990) give the following theoretical intuition, if each expert makes errors *independently* and its error rate is lower than 0.5, then the error of the ensemble model decreases *monotonically* with the number of included experts.

In the following, we will review ensemble learning based on the way that experts are combined (*architecture*) and how the prediction of each expert is used for the final ensemble prediction (*voting*), using the survey work Gomes et al. (2017). Concerning the ensemble architecture, we distinguish three types, parallel, cascading and hierarchical. In a parallel architecture, initially each expert predicts, then the outputs are combined together by a single classifier (for example a simple linear function) to create the final prediction. Cascading architectures include all arrangements at which the prediction of an expert is used as input to a series of classifiers (more than one). The last type of architecture is the hierarchical, where the prediction of each expert are combined based on a tree-like structure.

As far as voting is concerned, there are five categories of common mechanisms: majority, weighted majority, rank, classifier selection and relational. These categories can be ordered by their implementation complexity. Majority voting is the simplest where the majority of experts prediction is used as output of the ensemble. Instead of using equal weights for the majority computation the weighted majority scheme

uses a different weight value for each expert. In the rank category, predictions over all possible class labels are averaged over the experts, with different weights per expert, to compute the final prediction of the ensemble. Classifier selection mechanism follows the distinct weighting of each expert as the weighted majority one, but it changes these weights dynamically through the training process. In the last and most sophisticated voting category, experts can relate their prediction to other experts predictions, through a specified model, to finalize their prediction. Then their finalized prediction can be combined through a majority or weighted majority voting.

2.1.5 Prediction with Expert Advice

Based on the foundational book by Cesa-Bianchi and Lugosi (2006), we present a brief introduction in the field of prediction with expert advice. The basic scheme of this research area is a series of sequential decisions by a forecaster. That is, the forecaster goal is to guess an unknown sequence of $y_1, y_2, \ldots$ of an outcome space $\mathcal{Y}'$. These predictions $\hat{p}_1, \hat{p}_2, \ldots$ belong to the decision space $\mathcal{D}'$. The forecaster computes its predictions in one-after-the-other (*sequential*) fashion and we compare its predictive performance to the respective performance of a set of reference forecasters known as *experts*. That is, at each time point t the forecaster is informed for the set of the predictions of experts $\{f_{E,t} : E \in \mathcal{E}\}$, with $f_{E,t} \in \mathcal{D}$ and $\mathcal{E}$ be the fixed set of indices for the experts. The forecaster using the experts predictions $f_{E,t}$ outcomes its prediction $\hat{p}_t$ for the element in the next time-point y_t. After its guess the real value,y_t, is uncovered. The predictive performance is computed through a non-negative loss function $l : \mathcal{D}' \times \mathcal{Y} \to \mathbb{R}$.

This basic sequence of steps can be seen as a repeated game of two actors, at which for a given time-point t, the "environment" selects the new outcome y_t and the expert advice $\{f_{E,t} : E \in \mathcal{E}\}$. Then, the "forecaster" constructs its own guess $\hat{p}_t$ and subsequently the "environment" uncovers the true value of the current time-point $y_t \in \mathcal{Y}'$. In the final step of this prediction round, the loss for the forecaster and each expert is computed, $l(\hat{p}_t, y_t)$ and $l(f_{E,t}, y_t)$ respectively. We present the pseudo-code of this repeated game in the following Algorithm 1.

Algorithm 1 Prediction with Expert Advice

for each round $t = 1, 2, \ldots$ **do**
(1) the environment chooses the next outcome y_t and the expert advice $\{f_{E,t} : E \in \mathcal{E}\}$ and the forecaster is informed for the expert advice
(2) the forecaster chooses the prediction $\hat{p}_t \in \mathcal{D}'$
(3) the environment reveals the next outcome $y_t \in \mathcal{Y}'$
(4) the forecaster experiences loss $l(\hat{p}_t, y_t)$ and the expert E experiences loss $l(\hat{p}_t, y_t)$
end for

The main goal of a forecaster is to be as close as the best expert in predictive performance and thus the forecaster tried to minimize the *cumulative regret* (or regret)

with respect to each expert. The regret for an expert E is defined as follows:

$$R_{E,n} = \sum_{t=1}^{l}(l(\hat{p}_t, y_t) - l(f_{E,t}, y_t)) = \hat{L}_l - L_{E,l}, \tag{2.11}$$

where $\hat{L}_l = \sum_{t=1}^{l} l(\hat{p}_t, y_t)$ is the cumulative loss of the forecaster and $L_{E,l} = \sum_{t=1}^{l} l(f_{E,t}, y_t)$ to denote the cumulative loss of the expert E. Therefore, in the Equation 2.11, we defined the difference between the forecaster total loss and that of the E expert after l rounds of predictions. Besides, we can define the regret for only the specific time point t by:

$$r_{E,t} = l(\hat{p}_t, y_t) - l(f_{E,t}, y_t). \tag{2.12}$$

Therefore we can have that $R_{E,n} = \sum_{i=1}^{l} r_{E,t}$. Intuitively, we can regard the $r_{E,t}$ quantity as the regret that the forecaster experiences not having listened the expert E right after the reveal of the true outcome, y_t.

Given a loss function with specific properties, the work by Kivinen and Warmuth (1999) has proved that a forecaster which keeps exponentially decreasing weights for each expert E_i and uses these weights to compute the weighted average of their predictions as his final prediction ($\hat{p}$) can achieve cumulative loss that is bounded from above by the cumulative loss of the best expert plus a constant. This forecaster is known as the *Weighted Average Algorithm* (WAA). We present the pseudo-code of WAA as follows:

Algorithm 2 Weighted Average Algorithm (WAA) for combining expert predictions

Initialize the weights to some probability vector $v_{1,i}$; set the parameter c to some positive value
for each round $t = 1, \ldots, l$: **do**

1. Receive the instance x_t
2. Output the forecaster prediction $\hat{p}_t = \boldsymbol{v}_t \cdot \boldsymbol{x}_t$
3. Receive the outcome y_t
4. Update the weights by the *loss update* defined as follows:

$$v_{t+1,i} = v_{t,i} exp(-L(y_t, x_{t,i})/c)/norm_t$$

where

$$norm_t = \sum_{i=1}^{n} v_{t,i} exp(-L(y_t, x_{t,i})/c).$$

end for

The WAA needs to use a constant $c \geq c_L$. Given a loss function L, let us denote $L_y(\hat{y}) = L(y, \hat{y})$ for convenience in writing derivatives with respect to $\hat{y}$. To define c_L, we first need to define the quantity:

$$R(z, p, q) = \frac{L'_p(z)L'_q(z) - L'_q(z)L'_p(z)^2}{L'_p(z)L''_q(z) - L'_q(z)L''_p(z)}; \tag{2.13}$$

where we define $R(z, p, q) = 0$ in the special case $p = q$. Then the c_L is defined as:

$$c_L = \sup_{0 \leq z,p,q \leq B} R(z, p, q). \tag{2.14}$$

For a $y \in [0, B]$ and the square loss function (Equation 2.2), we derive $c_L = \frac{B^2}{2}$.Last we present the theorem on the bound of the WAA cumulative loss as stated by Kivinen and Warmuth (1999):

Theorem 1. *Let L be a monotone twice differentiable loss function and WAA be initialized with uniform initial weights $v_{1,i} = 1$ and with $c \geq c_L$. Then for any sequence $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l))$ holds:*

$$Loss_{WAA}(S) \leq (\min_i Loss_{\mathcal{E}_i}(S)) + cln(n).$$

2.2 Representation Learning

2.2.1 Vector Space Model

Here, we will introduce word embeddings mainly based on Almeida and Xexéo (2019). Word embeddings is the new established form of text representation used by Natural Language Processing (NLP) research community. Historically, the first used form of text representation was the Vector Space Model (VSM) introduced by Salton et al. (1975) and it was founded by Information Retrieval (IR) research community. This model represents each distinct term of a document as a vector in a T-dimensional space, where T is the number of words of the vocabulary of the used language. The values of each dimension of this T-dimensional vector can be binary, also known as *one-hot representation*, or real. The VSM facilitates the operations on term vector through the usage of linear algebra and statistics, inherited by the vector space model. For example, the similarity of two term can be calculated by the inner-product of the terms vectors.

The vector space model was used extensively by the IR and NLP communities approximately up to 2010. The main reasons that this model is now used only as a baseline comparison are its two main drawbacks. First, the model is not space-efficient, because the dimensions of the space are analogous to the dimensions of the language dictionary, not allowing efficient processing of very big data set of documents. Second and most important is the inability of the model to capture any linguistic feature, syntactic or semantic, of the used language. Therefore, a classifier, which uses such representation for the input words, cannot know the meaning of semantically similar words compared to the training set words. In turn, this will decrease the classifier performance for documents with unseen words. These drawbacks along with technical improvements of learning neural representation of words founded a new form of words representation known as word embeddings.

2.2.2 Word Embeddings

Harris Harris (1954) has suggested that words with similar meaning appear in similar contexts. In linguistics, this is also known as the *distributional hypothesis*. All methods to build word embeddings are stemming from this assumption. Consequently, we define the *word embeddings* as dense distribute and fixed-length word vectors, computed by word co-occurrence statistics based on the distributional hypothesis. The first creation of word embeddings is attributed to Miikkulainen and Dyer (1991) and Bengio et al. (2003). In the following years after the latter publication researchers have improved the time-efficiency of the neural network, which learns the word embeddings, and they have focused on designing models to build only word embeddings, without training a more general language model. Among these methods, the studies by Mikolov and colleagues Mikolov et al. (2013c,a,b) resulted to the first established version of word embeddings used by NLP community from 2013. Their methods are described in the following paragraph.

Mikolov and colleagues Mikolov et al. (2013a) have introduced two models to build word embeddings, the *continuous bag-of-words* (CBOW) and the *skip-gram* (SG). Both models are consisting of an one layer neural network followed by an non-linear output layer. The difference of the models is on the problem formulation. That is, in the CBOW model, the input is the surrounding words appearing before and after a center word and network aims to predict this center word. In contrast, SG model inputs the center/target word and predict its surrounding words. Words embeddings produced by these models, were the state-of-the-art word representation for NLP community from 2013 and at least 5 years. These dense word vectors resulted to state-of-the-art text classification systems. Another benefit of this representation is its ability to capture not only syntactic but also semantic similarities between words.

Even since, the adaption of words embeddings, researchers have investigated their ability to capture words semantic and syntactic information. We will present the most evident example of such evaluation methods. First, qualitative methods have been used word analogies, for example between names of countries and their respective capitals to show that the trained word embeddings can capture the semantic similarity of words Mikolov et al. (2013a). That is, if we search for the trained vector $X : X = v(Italy) - v(Rome) + v(Cairo)$, using cosine similarity, we will find that $X = v(Egypt)$, proving that these low-dimensional vector can capture the analogous similarity of related word pairs. Besides researchers have also applied quantitative methods to evaluate the captured linguistic information by the word embeddings. For instance, researchers have examined if the trained embeddings can be used to retrieve semantically similar words to a given query word compared to human annotated similar words of the query word. To do so, they commonly compute the correlation coefficient between the two resulted sets, the retrieved words found as similar to the query based on the trained embedding space and the human annotators respectively Mikolov et al. (2013b).

Word embeddings offer a time-efficient computation of word representation, but still they have shortcomings. The main two challenges with these traditional word embeddings models is their inability to produce context-aware and lexically inductive word representations. Concerning the first challenge, take as example the word "bank" based on its context it can have similar meanings (the financial institute or the land at the side of a river) SG and CBOW will learn only one vector representation for the word "bank". However, more sophisticated language models, such as ELMo Peters et al. (2018) and BERT Devlin et al. (2018), will learn a distinct vector for this word based on its context. There exist also methods to learn for a sequence of letter of a given word, n-grams, thus allowing to represent input words which are not seen in the training corpora, based on the learned vectors for words morphologically similar to these input words. Such example word vectors compose the FastText embeddings developed by Bojanowski et al. (2017).

2.3 Time Series

In the previous sub-chapter, we gave the foundation for machine learning methods which are used for static data sets. This means that the feature value of a instance do not change over time. Nevertheless, in many real-life scenarios variables change over time. Storing the sequence of values of a random variable X is known as $time-series$ and a common notation is using the natural number as time-index of the sequence:

$$X_t = x_1, x_2, \ldots, x_{t-1}, x_t$$

Two common applications on time series are forecasting and pattern identification. Fundamental concepts for both applications are the *trend* and *seasonality* notions. The first captures the linear or non-linear function that generates the values of the series in a general and systematic way. The second is the part of the series that generates the values by duplicating itself in periodic fashion.

2.3.1 Trend

Focus on long-term patterns or cycles and reduce the effects of short-term pattern, the technique of moving average is used. The simple moving average of the last n time points is defined as:

$$MA_t = MA_{t-1} + \frac{x_{t-(n-1)}}{n} + \frac{x_{t+1}}{n}.$$

2.3.2 Seasonality

Autocorrelation is the relation of a time-series with itself. It can be used to identify repeating patterns such as cycles of a same sequence of values. Let l to denote the

time lag between the current time point t the respective next one $t + l$, then the autocorrelation of these two points can be calculated as:

$$r(x, l) = \frac{\sum_{i=1}^{n-l}(x_i - \bar{x})(x_{i+l} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

where $\bar{x}$ is the mean of the first n values.

2.3.3 Stationarity

Many methods for prediction and pattern identification on time series data assume that the series is stationary. That is the mean, variance and autocorrelation of the series is *static*, it does not change over time.

2.3.4 ARIMA

Auto-Regressive Integrated Moving Average (ARIMA) models are a general class of models to capture seasonality, trend and even stationarity, by differencing, properties of a time series. Therefore, ARIMA methods can differenciate a times series and then cast it as a two-component function composed by the signal and the noise parts. Formally, given a time-series X_t where t is the time index and $x_i \in \mathbb{R}$, the ARIMA model is defined as follows:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_t, \tag{2.15}$$

where L is the lag operator, ϕ_i the parameters of the auto-regressive part of the model and θ_i the parameters of the moving average part of the model. Equation 2.15 is commonly denoted in short as ARIMA(p,d,q).

2.4 Data Types

2.4.1 Textual Streams

Nature of Data

Input data can be grouped by their change over time in: *stationary* and *transient*. Data of the first category are sampled from a stationary distribution, for example when a person enters images of the 0-9 digits as input data into a computer. However, nowadays many of the input data are collections of the same data type over time, also known as *data streams*. Examples of data streams are the posts accumulated

over time on a social media platform and the plant consumption measurements of a energy network collected in a specific frequency over time.

Automatic feeds of social media create textual data streams, collections of texts over time. These streams of textual posts have some main characteristics. That is, they are created with high speed, in large volumes and their enclosed information may vary over time, for instance change of public view on global warming. Therefore, classical classification approaches which build a single static model over a finite number of data samples cannot cope with the nature of this data type. Instead, based on Gama (2010), the requirements on a *streaming algorithm* imposed by textual data streams are: (i) to use limited computation and memory resources (ii) to process each data sample only once (iii) to be capable of predicting in an *anytime protocol* (iv) to have the ability to forget older data samples.

The last requirement, the model to be able to remove obsolete data samples, is due to the time-varying data generating distribution, a phenomenon known as *concept drift* first described by Schlimmer and Granger (1986). A concept drift is a change on the data distribution that cannot be predicted ahead of time. Such change modifies the relation of the produced data and the target class to be predicted for each data instance. There is a diverse set of possible changes of the data distribution for an example a reoccurring or gradual change in the distribution. That is, a new theme in social media may appear and gain users attention in an abrupt fashion, an always increasing fashion, an oscillating fashion or reoccurring one.

Opinions on diverse themes can be incrementally found in different Internet sources from BBC online to Twitter, creating streams of opinionated documents. In these documents, users give their opinion on themes as products, institutes, services, social and political events. They usually use short to medium text messages where they express their opinion, by presenting their views on the overall impression (explicit opinion), by presenting their views for the aspects of the service or by comparing different services or by presenting their "thwarted expectations" (implicit opinion), first introduced by Pang et al. (2002), or discuss facts of the service (neutral opinion). Imposing even more challenges for the design of a model to classify such opinionated textual streams, the opinion-related polarity of words may change over time, for instance in a tweet post: "today's weather is very warm", the polarity of "warm" depends on the season of the year. Besides, they may change also in time-related context. For example, the tweet post: "the music is peaceful", the polarity of the "peaceful" depends on the current music festival type (Blues or Hip-Hop). Last, as other data stream types opinionated streams are dynamic and present concept drifts, for instance, people changing their opinion on a specific subject over time and words.

Challenges in Streaming Algorithms

The main challenge in online learning for data streams is the *concept drift* phenomenon. We will first explain this phenomenon from the mathematical point of

view and then we will give a real-world example, based on the survey work by Gama et al. (2014). Applying the Bayes rule according to Duda et al. (1973), a classifier h will use the following equation to predict the class $y \in \{1, \ldots, c\}$ of an instance $\boldsymbol{x}$:

$$p(y|\boldsymbol{x}) = \frac{p(y)p(\boldsymbol{x}|y)}{\sum_{y=1}^{c} p(y)p(\boldsymbol{x}|y)} \tag{2.16}$$

As the stream of data evolves over time, quantities of Equation 2.16 may change in an unexpected manner. These changes may caused by a real-world situation (increase of public interest in climate change) or a natural effect (sensor read-outs are not possible due to a heavy storm in the energy plant). Formally, we will say that a concept drift happens between time point t_0 and t_1 when:

$$\exists \boldsymbol{x} : p_{t_0}(\boldsymbol{x}, y) \neq p_{t_1}(\boldsymbol{x}, y), \tag{2.17}$$

where the p_{t_0} and p_{t_1} are the joint probability of a seen instance $\boldsymbol{x}$ and the target class y for the t_0 and t_1 time-points respectively. Using the chain rule which links the joint probability with the conditional and the prior, that is:

$$p_t(X = \boldsymbol{x}, Y = y) = p_t(Y = y)p_t(X = \boldsymbol{x}|Y = y) = p_t(X = \boldsymbol{x})p_t(Y = y|X = \boldsymbol{x}), \tag{2.18}$$

Equation 2.17 holds, based on Equation 2.18, under the following assumptions:

- the prior probabilities of classes, $p(y)$ may change,
- the conditional probabilities $p(\boldsymbol{x}|y)$ may change,
- and thus, the posterior probabilities of classes $p(y|\boldsymbol{x})$ *may* change over time affecting the final prediction.

We distinguish the types of concept drift by the realization of the previous facts:

1. When the posterior probability, $p(y|\boldsymbol{x})$ changes with or without alternation of the $p(x)$, the prior *on* the data generating function, $\mathcal{D}$. Such change affects the correctness of the predictive decision of the classifier and therefore is named as *real concept drift*.
2. When the distribution of data change but the posterior $p(y|\boldsymbol{x})$ is not altered, then the classifier can still use its existing model to predict the current instances, we refer to this drift type as *virtual drift*.

Real concept drift was firstly described by Salganicoff (1993) and virtual drift by Delany et al. (2004).

To explain concept drift terms take as real-world example the search for a means of transport (bike or car). The tasks of a classifier is to read user's news feed and

predict if the user would be interested to read a post (y="relevant") or not (y="not relevant"). If the regulations of the city, where the user lives, have changed to promote the advertisements of used means of transport compared to brand new ones, but the news editor and the user interests have remained the same, the prior probabilities of the classes have experienced a drift. If the advertisement posts have started to describe the ecological footprint of the described vehicle, the advertisement posts on means of transport are *still relevant* for the user. This is an instance of virtual drift. In contrast, if the user has made his choice for an used bike and then he is now interested in his local city news, the advertisement posts in bikes and cars would turn now into *irrelevant* and the news posts about his local city will become *relevant*. Such scenario describes a real concept drift.

We can categorize the drift by the speed of change into five general class. A stream can experience a *sudden* drift caused by an abrupt change of temperature read-outs from a sensor during a storm. A sensor that starts to malfunction due to its time usage will introduce an *incremental* drift, because its read-outs will slowly be inaccurate. A newly installed sensor will slowly and steadily report correct measurements, giving an example of *gradual* drift. Last, old topics (climate change) may re-appear as popular in the Twitter stream, creating an example of *reoccurring* drift.

Streaming Algorithms Dealing with Concept Drift

Here, we review streaming algorithms dealing with concept drift phenomenon. These approaches can be grouped by their focus on parts of the learning model: memory, change detection, learning procedure and loss estimation. For the first category, *memory*, the initial methods were able to store only the current instance on the model. An example of such methods is the Very Fast Decision Tree (VFDT) by Domingos and Hulten (2000) which has used the Hoeffding bounds Hoeffding (1994) to optimally choose the number of instances needed to split a node of a decision tree; which is built using the information gain criterion. Subsequent methods design a forgetting mechanism, to remove obsolete instances from the model memory. The models may forget instantly by using a fixed-size sliding window Babcock et al. (2002) or gradually by keeping all instances in memory but decreasing their importance weights for the learning model Klinkenberg (2004). Besides, methods have focused on the *change detection* component by designing statistical methods to identify a possible drift in the data. A common example of this category is the ADaptive sliding WINdow (ADWIN) detection algorithm. In short, ADWIN keeps a *sliding* window of the most recent instances and a past window containing a higher amount of instances, then it applies the Hoeffding bounds to spot distribution changes between these two windows. If such difference is identified, the model is alarmed for a possible drift. Thus it will re-build its model for the current window of instances.

Streaming algorithms with focus on designing a sophisticated *learning model* to deal with concept drift may re-train the model, *at each time*, a new batch of data ar-

rives Gama et al. (2004). In contrast, incremental methods will process each instance at its arrival time and it will change its own stored sufficient statistics based on each instance. One established example of incremental methods is the CVFDT Hulten et al. (2001). Furthermore, models can adapt slowly to the drift, known as *blind* adaptation, or more quickly using an *informed* strategy most often coupled with a detection method. During an informed adaptation of the model, after the drift detection either the full model is re-build or only the part of the learned model affected by the drift, equivalently the affected subspace of the constructed hypothesis of the model. The CVFDT follows such local replacement technique. Researchers have also proposed ensemble learning techniques to overcome drifts. The intuition is to use each weak learner of the ensemble to follow a distinct concept distribution and then combine their based on their accuracy on the current instances of the stream. Such example is the SEA ensemble algorithm which trains a new weak classifier for each batch of sequential instances. This model predicts by simply averaging the predictions of the existing weak classifiers. The SEA ensemble *forgets*, by deleting the weak classifier with the minimum performance at each batch iteration.

In Chapter 3, we will describe our contribution on classifying textual streams with a specific type of drift, the *feature drift*, using an ensemble learning technique.

2.4.2 Protein Sequences

Nature of Data

Our second contribution is in the field of protein classification. We will give a brief overview on proteins based on Van Vranken and Weiss (2018). *Proteins* are macro-molecules playing important role for most of the biological processes. In 1983 Gerrit Jan Mulder, Dutch chemist, coined this name originated by the Greek word, πρώτειος (proteios), which means of first significance, to show their central part in biology. The elementary building blocks of proteins are the 20 amino acids, oligomers, which are connected by specific chemical bonds, "amide bonds". These formations result to molecules with low molecular mass, *peptides* or intermediate mass (multi-peptides). This chain of connected amino acids creates the protein backbone. Based on the backbone chemical properties, proteins fold in the three dimensional space and acquire well-defined substructures. Based on these sub-structures proteins interact with other macro-molecules or proteins and create *protein complexes* to accomplish or regulate a biological process. The hierarchical levels of protein structure are shown in Figure 2.1. Therefore, the most prominent question of structural biology is how to predict the protein structure based on its amino acid sequence.

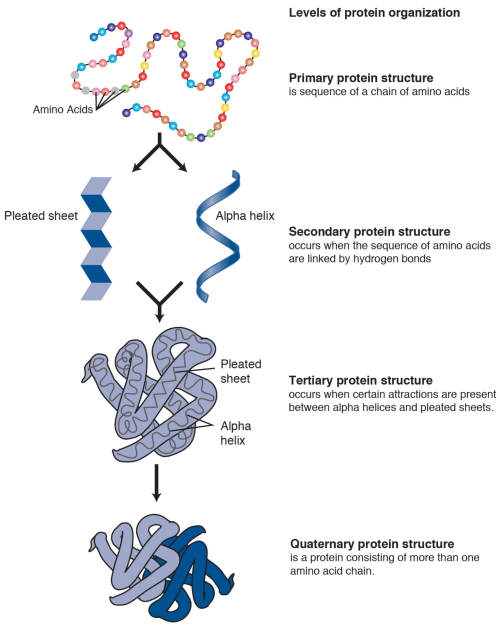


Figure 2.1. Hierarchical elements of protein structure. Adopted by National Human Genome Research Institute (NIH) resource page: NIH - Genetics Terms - Protein.

As proteins act based on their structure in space, a equivalently important research question is how to predict *protein function* based on the mere amino acid sequence of

the protein. The gold-standard method is the manual annotation of the target protein by domain experts, *bio-curators*. The bio-curators need to review publications related to the protein and the already determined structure of close-related proteins to elucidate the biochemical properties, the local and global structure and the main function of the target protein. Last step of this manual annotation is a specific experiment, *crystallography* experiment Hofmeister (1890); Osborne (1894), to determine the three-dimensional structure of the target protein, so as to cross-validate the manual annotation. However, all the steps of the manual annotation are time-consuming and needed extensive amount of resources. For that reason, databases containing manually annotated and/or reviewed proteins are needed. UniProt Knowledge Base (UniProtKB) is composed by two sub-databases the UniProtKB/SwissProt database including manually curated proteins and the UniProtKB/TrEMBL database which contains computer-assisted annotated proteins. Approximately the 0.26% of TrEMBL proteins are not been manually reviewed yet (UniProt version: March of 2021) Consortium (2021).

One important annotation field on UniProtKB database is the domains of the protein entry. A *domain* is a protein sub-unit that can fold independently of the rest of the whole sequence. Therefore, a domain does have its own individual function. To maintain its function through evolution, the domain's structure is conserved and thus it exhibits conserved sequence segments. As in engineering multi-unit systems are constructed by combining smaller and independent modules, nature tends to re-use self-functioning modules to new biological machines (proteins) with the goal to enable new complex functions. That is, it commonly accepted that proteins evolve also through re-arranging already self-functioning sub-structures (domains) Moore et al. (2008). This means that evolutionary signals can be found, between distinct proteins, not only by the presence of the same domain, but likewise by the existence of a similar *domain architecture*, the linear order of the domains found in this pair of proteins. Works by Moore et al. (2008); Forslund and Sonnhammer (2012) have presented detailed surveys on protein domains and the evolution of the architectures respectively. A common database for protein domains is InterPro Blum et al. (2021), which automatically integrates protein domains from a diverse set of biological databases.

Challenges in Classifying Protein Data

Machine learning approaches have been applied for various supervised learning tasks in protein research. Such techniques have used to predict protein biochemical and biological properties, thus to facilitate protein design and engineering, see Fox et al. (2007); Bedbrook et al. (2017). Nevertheless, classical ML methods have used protein properties for its input features. Besides, the number of such training examples, containing protein properties as features, should be sufficient large in order to build a robust learning model. Based on the high cost and execution time of laboratory experiments to measure such protein proteins, these two requirements for protein

learning introduce a fundamental challenge. To add on, another hurdle on data collection for protein learning, is to distinguish the informative protein properties, from the uninformative ones, *before* building the model. Moreover, deep learning techniques have also been applied in protein learning. Consequently, their need for large labeled data sets increases the significance of the large training set challenge. This is because, to assign labels for a large training set (in the order of 100,000 instances) results to high time and monetary cost.

Classifying Protein Data Using Representation Learning

Compared to extracting protein properties as features, using the mere protein sequence can be an effective answer on the aforementioned challenges on supervised learning for proteins. The first reason is the existence of high throughput laboratory techniques to read-out the protein sequence, compared to the time-consuming techniques to extract protein properties. Second and foremost, representation learning techniques for input have shown promising results in the NLP research domain. In short, representation learning tries to find an input representation, which the learning model will use to capture discriminating characteristics of the input and thus be more robust on input variation Bengio et al. (2013). The ultimate goal of representation learning is to improve prediction performance. Therefore the use of this sub-field of machine learning can enable an enhanced set of features for the protein instances lowering the need for biochemical properties as features and large labeled training sets.

Indeed, learning neural representations amino acid sequence of the protein instances has achieved great success, as shown in the example works: Asgari and Mofrad (2015); Yang et al. (2018); Alley et al. (2019); Rives et al. (2021). In ProtVec work Asgari and Mofrad (2015), the authors split the protein sequence in 3-mers with the 3 possible starting shifts. They then used the skip-gram model to learn the embeddings for each distinct 3-mer. The embedding for a protein was taken as the mean of all its 3-mers. Next, Yang et al. (2018) applied ProtVec by using doc2vec method Le and Mikolov (2014) to aggregate the embeddings of all 3-mers contained in a given protein into its whole embedding vector. Following the NLP trend to use models of higher-complexity to learn embeddings for input text, Alley et al. (2019) have deployed a language model with recurrent neural networks (RNN) to learn the embedding vector for each amino acid of approximately 24 million protein sequences (UniRef50 data set of UniProt Consortium (2021)). Therefore, the learned vectors capture properties of amino acids for a large set of protein families. Subsequently, the authors applied a *fine-tuning* step, by updating the embeddings with a trained labeled set targeting their specific protein family of interest. As final step, they benchmarked the predictive performance of trained models using these embeddings by predicting the stability and phenotype of diverse variants of the green fluorescent protein.

Besides, Rives et al. (2021) researched if application of a language model with

higher complexity, on a diverse set of sequences, could build representations capturing more protein characteristics. To do so, they applied the Bert language model on 250 million proteins (Uniparc data set of UniProt). Their experiments have shown that the learned representation capture the secondary structure and residue-residue contacts. To benchmark on downstream tasks, they compared their prediction performance on pathogenicity of protein variants and reported comparable results to the state-of-the-art variant predictors. All previous methods have shown the applicability of NLP techniques in protein learning. Nevertheless, researchers have designed methods to learn sequence embeddings using neural techniques customized for biological sequences. For example, Lu et al. (2020) have developed a model to learn representations for sub-sequences of a protein by differentiate between random and original fragments of a given protein. They have shown that this model achieved comparable performance with NLP models for proteins, with the beneficial need of *only* 2-10% of the complexity of the NLP models.

Our contribution on protein embeddings, Chapter 4, is the computation of protein domains embeddings, *dom2vec*, which can be *quantitatively* evaluated compared to sequence embeddings. We also have benchmarked the performance gain by the usage of *dom2vec* embeddings on down-stream prediction tasks, compared to sequence embeddings.

2.4.3 Genomic Variants

Nature of Data

Another foundational macromolecule for living organisms is the deoxyribonucleic acid (DNA). It is stored in the cells of all living organisms. Compared to proteins, DNA is a double-stranded chain of four chemical substructures, *nucleotides* or *bases*: adenine (A), guanine (G), cytosine (C) and thymine (T). The DNA is fundamental for living organsisms, as the *sequence* of its alphabet encloses all information required for a cell to develop, grow, function and reproduce. More specifically, based on the central dogma of Biology, certain DNA regions, *genes*, encode the instructions to create (*transcribe*) ribonucleic acid (RNA) molecules, which in turn are translated into proteins, the basic biological "machines" of the cell. This information is passed to the offsprings of parent organisms forming the majority of hereditary information between generations. The three-dimensional structure of the DNA was discovered by Watson and Crick in 1953 Watson and Crick (1953). The verified double helix structure of DNA is shown, as cartoon, in Figure 2.2. In eukaryotes the DNA is stored and maintained in the nucleus compartment of the cell. Inside the nucleus, the DNA is grouped into 23 large units, *chromosomes*, firstly identified using microscopy experiments by Schleiden Schwann (1847), Virchow and Bütschli in the second half of the 19th century.

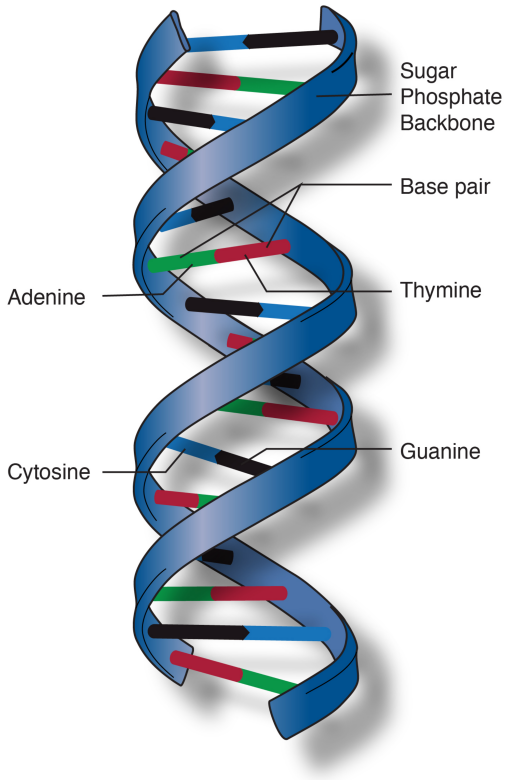


Figure 2.2. Double helix of DNA. Adopted by National Human Genome Research Institute (NIH) resource page: NIH - Genetics Terms - Double Helix.

Based on its importance for life, researchers aimed to compile the complete collection of DNA bases for a given cell, *genome*. Frederik Sanger and colleagues achieved to read short segments of the genome, through a technique known as *Sanger sequencing* in 1977 Sanger and Coulson (1975). A international consortium, named Human Genome Project (HGP), sequenced the 99% of the human genome, containing 3.2 billion bases, in 2003 Collins et al. (2003), at an adequate error level to be used in research purposes. This draft genome served as a *reference* to find differences, *variations*, in the genetic code between individuals. It is estimated that the genomic variants of a subject compared to the human reference is approximately 0.1%. This genomic variation is sufficient to make each of us unique, because a smaller set of these genomic variants is related to physical traits, such as height or eye color. Nevertheless, the genomic variants also cause disease phenotypes, *genetic diseases*. Currently are cataloged 6,000 genetic diseases Hamosh et al. (2000). The most common types of variants are shown in Figure 2.3.

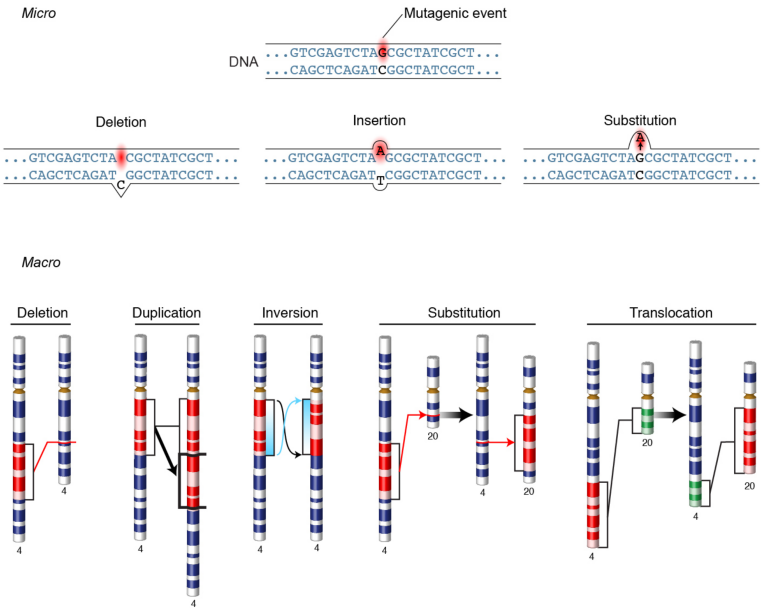


Figure 2.3. Types of DNA Variants. Adopted by National Human Genome Research Institute (NIH) resource page: NIH - Genetics Terms - Mutation.

As a consequence of knowing that a large amount of diseases is based on genomic variation, diagnostic laboratories would benefit from sequencing the whole genome of each individual. However, the use of Sanger sequencing to read-out the whole genome of an individual was time-consuming and costly. In 2005, a new technology for sequencing was invented, *Next Generation Sequencing* (NGS), which allowed the high-throughput reading of the genome with low cost and sufficiently low error rate. From 2015, the NGS was established in the level to be used in routine diagnostic setting to assist the clinicians Singh et al. (2016).

After the NGS technique extracts a magnitude of copies of short reads-out of each base on the genome, these reads are *mapped* to human reference and differences between the observed (patient's) sequence and the reference are called. These differences are also known as *variants*. Variants can be categorized by their length in micro- and macro-variants. Micro-variants are short differences between the observed and the reference and include the following types: (i) *substitution* of a reference nucleotide with another one, (ii) *deletion* of a reference nucleotide and (iii) *insertion* of a base next to reference nucleotide. The macro-variants are larger differences between the observed and reference genomes and are categorized in five sub-classes: (i) substitution of a section of one chromosome with a section of another chromosome, (ii) deletion of a larger section of a chromosome, (iii) *duplication*, the insertion of the same sequence of a larger section of a reference chromosome to the same chromosome, (iv) *inversion* insertion of the inverted sequence of larger section of a chromosome

to the same chromosome and (v) *translocation*, the directionality-preserved exchange of a chromosome section with another chromosome section. All these variants are caused by naturally-occurring errors during DNA replication or by the interaction of a cell with external sources such special chemicals or viruses. Next, we discuss the challenges to classify the identified variants of a patient in the routine diagnostic setting.

Challenges in Automatic Interpretation of Genomic Variants

In the routine diagnostic setting, a patient with a given phenotype is sequenced and its variants are extracted using the previously described pipeline. However, there is still the need to interpret these variants, before the diagnostic personnel can advise the patient, finishing the clinical genomics analysis cycle. Most of the times, to interpret a variant we need to classify its *pathogenicity* which can be (i) benign, (ii) pathogenic or (iii) uncertain significance. Given that patient's variants found in human genes are approximately 60,000 the manual curation of all variants by a diagnostic personnel poses the two following challenges.

First, the human curator should annotate each variant, using bioinformatics software and by reviewing biological articles with functional studies on this variant. Next, using these annotation data, the human expert can classify the potential result of the variant. This results to a time-consuming and error-prone step of the interpretation process. Second, as each diagnostic laboratory may use distinct biological criteria to classify a variant, there is the danger of inconsistency on the classification of a specific variant. Therefore, methods to classify variants without the need of the manual curation are needed.

There exist two general approaches on designing a variant classification algorithm. The first, uses the machine learning paradigm to learn a prediction hypothesis based on already classified human genomic variants. Three examples of machine learning classifiers for variant pathogenicity are: CADD Rentzsch et al. (2019), REVEL Ioannidis et al. (2016) and LEAP Lai et al. (2020). The latter set of approaches use an alternative method which it first examines accepted evidence-based criteria to assess the effect of a given variant based on different type of data and then it applies a *rules-based* scheme to assign the final pathogenic class of a variant. Experienced molecular geneticists commonly prefer to trust classification algorithms of the second paradigm, because they can *firmly* review the evidence-based criteria, assigned by such algorithm and therefore grant the variant classification in routine diagnostic manner. As a consequence, accepted methods and guidelines are needed for the variant classification.

To fill this gap on diagnostic analysis cycle, the American College of Medical Genetics and Genomics (ACMG) and the Association for Molecular Pathology (AMP) have established a working group to create twenty eight evidence-based criteria to classify a genomic variant in one of the following give classes: benign, likely benign,

variant of uncertain significance (VUS), likely pathogenic and pathogenic Richards et al. (2015). These criteria are described for a general phenotype stemming out of a hereditary disease trait. As far as the automatic implementation of these criteria is concerned, there are important obstacles to assess a variant using the ACMG guidelines. First, a molecular geneticist, curator, needs access to population, computational, functional and segregation data to use these evidence-based criteria. Besides, these evidences contain different degree of importance (ranging from very strong to supporting) and are attributed to the two broader impact classes, benign and pathogenic. After the human curator examines all evidences, she will combine the ones activated by a rules-based scheme also published by ACMG to classify the variant into one of the five pathogenicity classes. Even of the great benefit of the establishment of the ACMG guidelines, there is yet the unfulfilled requirement of algorithms which can apply these specifications *automatically*; whose classifications will be only reviewed by the diagnostic personnel before the final classification of variants.

Automatic Interpretation of Genomic Variants Using the ACMG Guidelines

Li and Wang (2017) implemented the InterVar, the first algorithm to automate the variant classification using the ACMG criteria. In more details, InterVar takes advantage of functional and computational annotations of variants by the ANNOVAR database Wang et al. (2010) to examine 18 out of the total 28 criteria, without the need of the human curator. The unimplemented criteria by InterVar need manual examination of functional, population and segregation data. A subset of the unimplemented functional criteria are covered by the VarSome search engine Kopanos et al. (2019), utilizing indexing techniques to extract publications with functional studies on the specific variant. In total, the VarSome algorithm implements 19 criteria out of the 28 ACMG criteria. Last, this search engine uses the users feedback, e.g. manual assessment of a criterion, to enhance the variant classification. The last example of variant classification algorithm focuses on automating the criterion that need manual assessment, such as criteria based on segregation and functional data. To this end, Mastermind algorithm Chunn et al. (2020) indexes the title, abstract, full text and supplementary material of medical publications registered in the National Library of Medicine/MEDLINE/PubMed database. After the indexing step, the algorithm associates several variants types to the functional information described by the indexed publications. The applicable variant types are single-base variants, copy number variants (CNV), deleted or duplicated stretches of 1,000 nucleotides or more, and gene fusion pairs; a pair of genes which, when are found next to each other, can express a hybrid gene. Applying these algorithms on a clinical diagnostics setting includes two obstacles: the free license only for academic usage and the closed source code.

Hearing impairment is the most common sensory disorder and a genetic etiology can explain half of affected patients Shearer et al. (2017). ACMG has refined their

guidelines for hearing impairment phenotype. Though, the closed source code of all these algorithms does not facilitate the creation of variant classification system for this disease. In Chapter 5, we describe our open source method GenOtoScope to annotate and classify variants associated with congenital hearing loss.

3 TemporalMNB: Learning under Feature Drifts in Textual Streams

In this chapter, we present our contributions by answering our first research question RQ1, defined in 1.2.1. Namely, how to adapt an established learning model for a special type of concept drift, *feature drifts*? How to ensure that this adaptation contain statistical guarantees compared to the feature best-describing process?

Huge amounts of textual streams are generated nowadays, especially in social networks like Twitter and Facebook. As the discussion topics and user opinions on those topics change drastically with time, those streams undergo changes in data distribution, leading to changes in the concept to be learned, a phenomenon called concept drift. One particular type of drift, that has not yet attracted a lot of attention is feature drift, i.e., changes in the features that are relevant for the learning task at hand. In this work, we propose an approach for handling feature drifts in textual streams.

Our approach integrates i) an ensemble-based mechanism to accurately predict the feature/word values for the next time-point by taking into account the different features might be subject to different temporal trends and ii) a sketch-based feature space maintenance mechanism that allows for a memory-bounded maintenance of the feature space over the stream. Experiments with textual streams from the sentiment analysis, email preference and spam detection demonstrate that our approach achieves significantly better or competitive performance compared to baselines.

3.1 Introduction

Huge amounts of textual streams are generated nowadays, especially in social networks like Twitter and Facebook. A key characteristic of those streams is velocity, i.e., the content of the stream changes over time as new topics arise, old topics

disappear and even for persistent topics, changes might occur. Such changes in the underlying data generation process might cause changes in the learned hypothesis, a phenomenon known as concept-drift Gama et al. (2014). Concept drift introduces new challenges for stream learners with the most important being the ability of the models to adapt to the underlying population changes. One particular type of change, that has not yet attracted a lot of interest Barddal et al. (2017) is feature drifts, i.e., changes in the features that are relevant for the learning task. The recent publication of Manzoor et al. (2018) discusses such streams referring to them as *feature-evolving streams*, where new features may arrive over time and existing features may update their value.

Such changes are especially frequent in textual streams as the feature space is high-dimensional (e.g., when considering words as features) and sparse (i.e., words are not observed in all documents and over the whole course of the stream). To deal with the new incoming features, one has to maintain a valid feature space over time that reflects the (temporal) importance of the features for the learning task. We mostly interested in the second property of such streams, the update of value of features. We are especially interested in temporal feature drifts, i.e., changes in the importance of features for the different classes over time, we refer to this as *trends*, hereafter. For the sentiment analysis task, we show an example of words with temporal trends in 3.1.

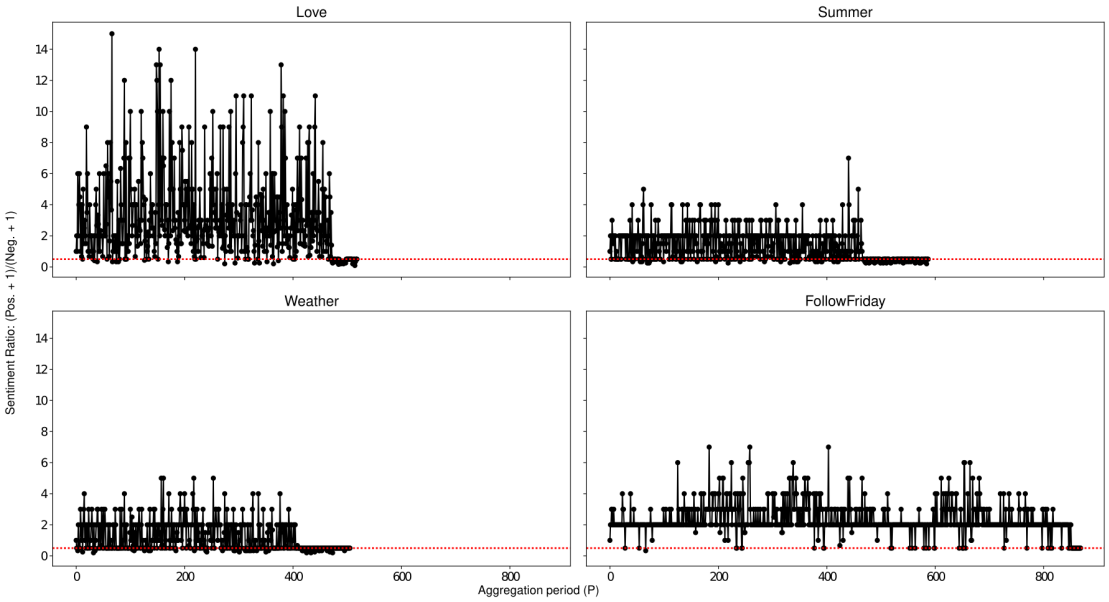


Figure 3.1. Sentiment Ratio ($\frac{\text{num. positives}+1}{\text{num. negatives}+1}$) for four words, where the red dashed line indicates the threshold for the change of sentiment.

By the figure we observe that exist words which are always associated with the same class like the word “love” (positive). On the other hand, there are words like

"summer" that display a seasonal trend and are probably negative out of season. Or words, like "weather" that display an autocorrelated trend depending on the weather conditions of the last days. Finally, there are *sudden* words such as "followFriday" that display short peaks in a class. Such sort of temporal drifts might also occur at the class level, more positive tweets in weekends as shown by Vosoughi et al. (2016). Such temporal trends are existing also in other textual streams such as in prediction of email preference or spam detection. For example an user may change her interest on a subject turning not interesting email to interesting ones as time evolves.

In this work, we propose an approach for handling feature drifts that discovers trends in the association of features with the different classes and uses those trends to predict the feature values for the next timepoint. As already shown via the aforementioned examples, there is a large variety of such trends from regular to seasonal and sudden occurrences; to capture this variety we propose an ensemble that takes into account different feature periodicities. The ensemble's prediction are seamlessly integrated in a Multinomial Naive Bayes (MNB) classifier whose model naturally depend on the class conditional feature probabilities. Although there are already approaches for adaptive MNB classifiers over streams, e.g., Nishida et al. (2012) and Wagner et al. (2015), our work is the first to *explicitly* tackle the feature drift problem. Our experimental findings show that such an explicit handling of feature drifts, result in improved performance compared to existing adaptive approaches.

The remainder of the paper is organized as follows: related work is discussed in Section 3. Basic concepts are discussed in Section 3, together with the problem definition. Our approach for handling feature drifts in textual streams by exploiting different feature/class trends is presented in Section 3. Experimental results are discussed in Section 3.4.3. Finally, we conclude in Section 3.5.5.

3.2 Related Work

Data streams are generated from non-stationary distributions and therefore, the learned hypotheses might change over the stream, a phenomenon known as *concept drifts* Gama et al. (2014).

A particular type of drifts, not adequately addressed thus far by the related work, is *feature drifts* referring to changes in the relation between a feature and the target class over time. The authors in Nguyen et al. (2012a) argue that concept drifts might lead into feature drifts and propose online feature selection to maintain a representative feature space over the stream, upon which a heterogeneous ensemble, of MNB and SVM experts, is trained.

Our approach also maintains a representative feature space over the stream, however differently from online feature selection methods our goal is to use feature history over the stream and different feature trend detectors in order to predict future feature

values that comprise the input to an final MNB model.

Feature drifts are frequent in textual streams; we focus hereafter on approaches based on Naive Bayes (NB) classifiers. In Katakis et al. (2006), the authors couple NBs with incremental feature selection based on information gain, for the email spam detection task. In Lebanon and Zhao (2008), a local likelihood method was presented that extends NBs by considering feature/word information from a particular time-spanning window. In Nishida et al. (2012), the authors employ exponential weighted moving average to improve NBs in the presence of sudden words. In Wagner et al. (2015), a temporal extension of NBs is proposed that decays the likelihood of a feature/word for a class and also the class priors based on the recency of their occurrence in the stream. In contrast to the aforementioned methods, in our approach we explicitly model feature drifts and moreover, we consider four different types of feature drifts.

Regarding the application domain, feature drifts for textual streams have been investigated in the context of sentiment analysis e.g., Wagner et al. (2015) and spam detection, e.g., Lin et al. (2017). Besides, time series prediction methods have been employed for analyzing sentiment in the Twitter stream. For example, in Giachanou and Crestani (2016), such methods are used to investigate particular sentiment characteristics like sentiment velocity for a given entity, whereas in Nguyen et al. (2012b) the goal is to predict the overall sentiment in Twitter stream. Finally in Melidis et al. (2018a), authors improved lexicon-based sentiment analysis using the predicted sentiment of words based on simple time series methods. In our approach, we also rely on time-series prediction methods for predicting different trends for the features/words, however those predictions are integrated in an MNB model for predicting sentiment at the tweet level.

Most similar to our work, is the taxi-demand prediction method of Moreira-Matias et al. (2013) which consists of an ensemble of three experts, each capturing different trends, namely, regular, seasonal and autocorrelated. In our approach, we moreover allow for the detection of sudden events and we average out the predictions of the experts assuring that the ensemble will predict as good as the best single expert or the best combination of experts in hindsight. Moreover, the application domain is completely different.

3.3 Basic Concepts and Problem Statement

We observe a textual stream D of documents arriving at distinct time-points $t_1, \cdots, t_i$ with t_i being the current time-point. Depending on the application, at each time-point t_i, a batch of documents instead of a single document might arrive. A document $d \in D$ is represented by the bag-of-words model as $d = \{v_1, v_2, \cdots v_k\}$. Given a predefined set of classes C, the goal of a stream classifier is to maintain a valid classification model over the stream. Without loss of generality, we assume

the classification problem is a binary classification problem, i.e., $C = \{+, -\}$. Our learning setting is fully supervised, that is, class labels are available for all documents. However, the label of a document d is available shortly after its arrival. Therefore, the goal is to make a prediction for d using the current classifier and then, upon the arrival of its label to use the labeled instance $(d, c), c \in C$, for training. This setup is known as first-test-then-train or prequential evaluation Gama (2010).

Assuming a (hidden) probability distribution P generating the instances of D, the characteristics of P might change with time, i.e., for two time-points $t_i \neq t_j$ it might hold that $P_i \neq P_j$, a phenomenon called *concept drift* Gama et al. (2014). There are different reasons for concept drifts in a stream environment, which can be explained based on the Bayes theorem Hoens et al. (2012). In particular, according to the Bayes theorem, the classification of a new document d from the stream depends on the class priors, i.e., $P(c), c \in C$, and the likelihood of observing d under the different classes, i.e., $P(d|c)$:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{3.1}$$

We assume that the document probability $P(d)$ is the same for all documents $d \in D$. Consequently, concept drift $P_i \neq P_j$ between two different time-points $t_i \neq t_j$ might occur due to changes in the following variables of Equation 3.1:

[C1] changes in the class priors, i.e., $P_i(c) \neq P_j(c)$.

[C2] changes in the likelihood of a document for a class $c \in C$, i.e., $P_i(d|c) \neq P_j(d|c)$.

[C3] changes in the posterior of a document for a class $c \in C$, i.e., $P_i(c|d) \neq P_j(c|d)$.

A specific type of drifts, not directly captured by the $C1 - C3$ types above, is *feature drifts* Barddal et al. (2017). A feature drift occurs when there are changes in the relevance of the features for the learning task over time. We distinguish between two types of feature drifts that might occur between two different time-points $t_i \neq t_j$, namely:

[F1] the likelihood of a feature v for a class c may change with time, i.e., $P_i(v|c) \neq P_j(v|c)$. For example, for the sentiment classification task the feature "summer" displays a seasonal trend being positive in-season and negative off-season.

[F2] the feature space might change, i.e., $F_i \neq F_j$, where F_i (F_j) is the feature space for time-point t_i (t_j, respectively). In literature this is referred as dynamic feature space, e.g., Nguyen et al. (2012a).

Our goal is to build a classification model over the stream that besides dealing with concept drifts (types $C1 - C3$), it also explicitly targets feature drifts (types $F1$ and $F2$). An MNB classifier comprises a natural model choice for dealing with feature drifts as it decomposes the problem of estimating the likelihood of a document for a

class to the problem of estimating the likelihood of observing the features/words of the document under that class (the so-called class-conditional feature independence assumption). In particular, according to a MNB classifier, the class of a document d is the class that maximizes the posterior, namely:

$$\hat{c}_d = \underset{c \in C}{argmax} P(c)P(d|c) = \underset{c \in C}{argmax} P(c) \prod_{v \in d} P(v|c) \tag{3.2}$$

Based on this formula, MNB captures $F1$ type drifts (variable $P(v|c)$) and $C2$ type drifts (by combining $F1$ type drifts as: $P(d|c) = \prod_{v \in d} P(v|c)$). Moreover, the classifier explicitly models the class priors, so it tackles $C1$ type drifts. Finally, it tackles $F2$ type drifts by maintaining a feature space of the top most frequent features over the stream (c.f. Section 3.4.2). Since, $C3$ type drift is the result of $C1$ and $C2$ type drifts, we show how to integrate $C1-C3$ and $F1$ type drifts to an MNB classifier in Section 3.4.3. The seamless integration of feature drifts in the model is not the only reason for choosing MNB as the learning model. The MNB classifier has proven to perform modestly over streams Bifet et al. (2009) and in particular, over high dimensional streams like textual streams Spiliopoulou et al. (2016). Moreover, such models can be efficiently maintained over streams as they rely on simple statistics.

3.4 Methods

We propose an MNB classifier that deals with concept drifts and feature drifts. Our approach is based on the observation that different features/words follow different trends with respect to their association to the class attribute. That is, in a sentiment classification task the word "love" has a *regular* trend in positive class, the word "summer" has a *seasonal* trend, the word "weather" may exhibit *autocorrelated* trend based on the atmospheric conditions of previous days and sudden words, in a Twitter stream, such as "followFriday" may exhibit short (*sudden*) peak trends. Therefore, our idea is to learn those trends for each feature/word independently and combine their predictions via an MNB classifier. An overview of our framework is shown in Figure 3.2.

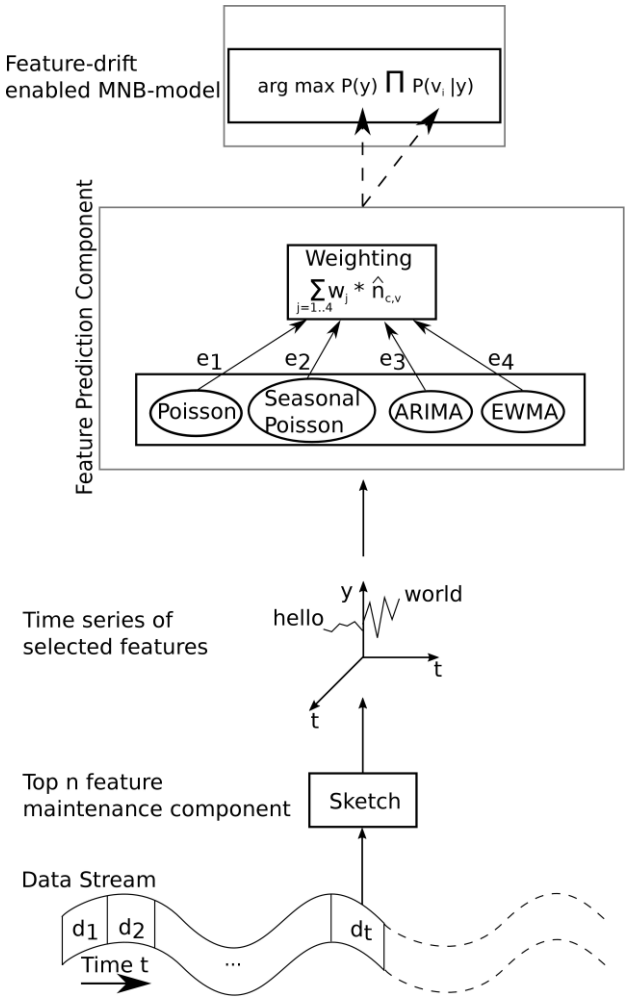


Figure 3.2. Overview of our approach.

As shown in this figure, our approach combines three components:

- a *feature prediction component* that uses an ensemble of experts each capturing distinct feature/word trends to tackle the $C1 - C3$ and $F1$ type drifts (Section 3.4.1),
- a *feature space maintenance component* that maintains a valid fixed size feature space over the stream to fulfill the memory-bounded requirement of streams but also to tackle $F2$ type drifts (Section 3.4.2), and
- a *feature-drift enabled MNB model* that utilizes the aforementioned components for document classification (Section 3.4.3).

Consequently, the proposed the feature-drift enabled MNB model can accommodate all types of drift $C1, C2, C3, F1$, and $F2$.

3.4.1 An Ensemble for Learning Different Feature Trends

The goal of the ensemble is to predict for each feature/word its value at the next time-point. To this end, we rely on a feature's history from the stream. In particular, let us assume the textual stream $\mathbf{D}$ arriving over time and let $T = \{t_1, .., t_i\}$ be the ordered set of observed time-points up to point t_i. We observe feature occurrences over the stream and in particular, class priors and word class-conditional counts. For simplicity, hereafter we refer to words, however we follow the same methodology for class priors.

We denote $n_{v,c}$ the conditional count of a word v in a class $c \in C$. Assuming an aggregation period of P time units, the sampling values form a discrete univariate time series, defined as follows:

$$N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\} \tag{3.3}$$

where $n_{v,c}(lP)$ is the observed conditional count of v in class c at the l-th aggregation period. We will refer to the history of the whole time series as *word history*, H_v. Each of the following models will use the whole or a sliding window of H_v, which is referred to *model history*, hereafter.

In order to solve the problem statement, we will present an ensemble of experts each modeling a distinct trend. We present each expert in Section 3.4.1 and the aggregation of their prediction by the ensemble in Section 3.4.1.

Feature Prediction by Experts

Thus far we discuss four distinct trends, *regular*, *seasonal regular*, *autocorrelated* and *sudden*. In this section we present four models, each to capture one of these distinct trends.

Poisson Model: We employ a *Poisson* model to capture *regular* trends. This model uses the *Poisson* distribution to relate the probability of having n as $n_{v,c}((l+1)P)$, $(P(n_{v,c}((l+1)P) = n))$ defined by the following equation:

$$P(n; \lambda) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{3.4}$$

where λ is the average number of $n_{v,c}(P)$ over the *model history*. The goal is to learn regular trends over the *whole* history of the word, therefore the *Poisson* model utilizes the whole *word history*. That is $H_{\text{Poisson}} \equiv H_v$.

Seasonal Poisson Model: This model can capture time series with a *seasonal regular* trend. Using the exponential smoothing algorithm Holt (2004), the expert will find the average over each group of aggregation periods and then it will aggregate all averages using weighted average with fading weights α_s based on the recency of the group. For example if the aggregation period is $P = 1$ hour and the

model group these periods in weeks, storing the last γ_s weeks, it will keep $7 \cdot 24$ aggregation periods for each group (week). Then it will aggregate the average of each week using the exponential smoothing algorithm. These weights increase as we move from the current week to the previous ones. By this, the averages observed two weeks before are less relevant to the average observed in *this* week. We calculate the weights set ω using the exponential smoothing algorithm Holt (2004). Thus we can compute the weighted average over the weeks of the history.

Thus we define ω_s as follows:

$$\omega_s = \alpha_s \cdot \{1, (1-\alpha_s), (1-\alpha_s)^2, \ldots, (1-\alpha_s)^{\gamma_s - 1}\}, \gamma_s \in \mathbb{N} \tag{3.5}$$

where γ_s is the number of groups of aggregation periods which the model considers and $0 < a_s < 1$ is the smoothing factor. Both variables are user-defined. Having computed the average for each group, $\lambda_k, k = \{1, \ldots, \gamma_s\}$, we can now define the weighted average μ_s as follows:

$$\mu_s = \sum_{k=1}^{\gamma_s} \frac{\lambda_k \cdot \omega_{s_k}}{\Omega_s}, \Omega_s = \sum_{k=1}^{\gamma_s} \omega_{s_k} \tag{3.6}$$

Then substituting the smoothed average, μ_s, into 3.4 we get the seasonal Poisson model. The expert keeps the newest γ_s groups containing g aggregation periods (P), consequently its history is sliding and equals to $|H_{\text{Seasonal}}| = \gamma_s \; g \; P$ and it holds $H_{\text{Seasonal}} \subset H_v$.

Autoregressive Integrated Moving Average Model: To model autocorrelated trends, we use the AutoRegressive Integrated Moving Average technique (ARIMA) Box and Jenkins (1976). The model is defined by three parameters $ARIMA(p, d, q)$, where p and q are the order of the autoregressive and moving average submodels, respectively and d is the lag of the series. The model stores a *sliding window* of size γ_a aggregation periods (P). We denote the model's history by $H_{\text{ARIMA}} \subset H_v$ and it holds $|H_{\text{ARIMA}}| = \gamma_a \; P$. A detailed description of ARIMA model can be found at Cryer and Chan (2008) (Chapter 4 and 5).

Exponential Weighted Moving Average Model: The last model intends to capture regularity only on the short-term history and it is inspired by Nishida et al. (2012). The model aggregates the observations in the history $H_{\text{EWMA}} \subset H_v$ using the exponential moving average with fading weights α_e reflecting the recency of the observations. More formally, we use the exponential smoothing Holt (2004) to weight each observed value ($N_{v,c}(jP)$) of the short history, containing the last γ_e aggregation periods P of the series ($H_{\text{EWMA}} \subset H_v$). The model keeps a sliding window over the last $\gamma_e \; P$, $|H_{\text{EWMA}}| = \gamma_e \; P$. Then predicts the future value as the weighted average of the values in H_{EWMA}, using the same logic as the second model. We define the set of weights w_e for each observed value inside the sliding window of size γ_e:

$$\omega_e = \alpha_e \cdot \{1, (1-\alpha_e), (1-\alpha_e)^2, \ldots, (1-\alpha_e)^{\gamma_e - 1}\}, \gamma_e \in \mathbb{N} \tag{3.7}$$

We combine the predictions using the ω_e of 3.7 as their weighted average μ_e as follows:

$$\mu_e = \sum_{k=1}^{\gamma_e} \frac{X_{i,t_k} \cdot \omega_{e_k}}{\Omega_e}, \Omega_e = \sum_{k=1}^{\gamma_e} \omega_{e_k} \tag{3.8}$$

Averaging Predictions of Experts

Our assumption is that the trend of each word can be modeled in the best way by one expert or by a (weighted) combination of them. More formally, given the time series $N_{v,c}$ up to aggregation period $l-1$, the M experts will predict for the next aggregation period l and input their values to the ensemble, which will aggregate their prediction to result in the final prediction. In this work, we have four experts ($M = 4$). For them we denote their predictions as $\hat{n}_{v,c}(l)^{\text{Poisson}}, \hat{n}_{v,c}(l)^{\text{Seasonal}}, \hat{n}_{v,c}(l)^{\text{ARIMA}}, \hat{n}_{v,c}(l)^{\text{EWMA}}$ and of the ensemble as $\hat{n}_{v,c}(l)^{\text{Ensemble}}$. Let $n_{v,c}(l)$ be the observed value of the conditional count of v in c, at the next aggregation period l. We assume each of the predicted values and the observed one are in the range of $[0, B]$, where $B \in \mathbb{N}$ is the upper limit of the conditional count of such word in the total stream and it is user-defined.

To measure the performance of the ensemble we use a *loss function* L which outputs a non-negative quantity $L(n_{v,c}(l), \hat{n}_{v,c}(l))$ indicating the difference between the observed (true) outcome and the predicted one. Assuming an expert that can best model the input then we can measure the difference of the ensemble performance compared to this learner using the regret measure, r, which is defined as:

$$r = \sum_{l} L(n_{v,c}(l), \hat{n}_{v,c}(l)^{Ensemble}) - \min_{\substack{e=Poisson,Seasonal,\\ ARIMA,EWMA}} \sum_{l} L(n_{v,c}(l), \hat{n}_{v,c}(l)^{e}) \tag{3.9}$$

In order to predict with guaranteed regret we follow the results of Kivinen and Warmuth (1999). To this end we use the weighted average algorithm, WAA, as formulated in Kivinen and Warmuth (1999). We show the whole method in algorithm 3, which follows the prequential process. We use the square loss function, $L(n_{v,c}, \hat{n}_{v,c}) = (n_{v,c} - \hat{n}_{v,c})^2$. By the equation 8 in Kivinen and Warmuth (1999), for $n_{v,c} \in [0, B]$ we compute the $\tilde{c}_L$ equal to $2B^2$. This weighting scheme gives us constant upper bound on the regret of the ensemble predictions, as stated by the following theorem:

Theorem 2. *[Kivinen & Warmuth Kivinen and Warmuth (1999)] Let L be a monotone convex twice differentiable loss function the WAA as in algorithm 3 with uniform initial weights, $w_{l=1,i} = 1$ and with $c \geq \tilde{c}_L$. Then for any sequence of $N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\}$ we have*

$$Loss_{WAA}(N_{v,c}) \leq (min_e Loss_e(N_{v,c})) + clnM. \tag{3.10}$$

As stated in Kivinen and Warmuth (1999), having a bound for the regret compared to the combination of learners that models best the input is better than regrets

compared to the best single model (2). Using the same setting, we can state the following theorem that bounds the regret compared to the best combination as follows:

Theorem 3. *[Kivinen & Warmuth Kivinen and Warmuth (1999)] Let L be a monotone convex twice differentiable loss function in WAA as in algorithm 3 using arbitrary initial weights, $\mathbf{q}_{l=1}$ and parameter $c = \tilde{c}_L$. Then for any sequence of $N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\}$ and for all probability vectors $\mathbf{p}$ we have*

$$Loss_{WAA}(S) \leq Loss_{\mathbf{p}}^{\mathbf{avg}} + \tilde{c}_L d_{re}(\mathbf{p}, \mathbf{q}). \tag{3.11}$$

Where $\mathbf{q}_t$ is the weight vector to combine each expert. As the weight vector is normalized and sums up to 1 is also a probability vector. Let p be such a probability vector, being the best combination of averaging the experts. To measure the difference of the two probability vectors, we use the *relative entropy* defined as $d_{\mathrm{re}}(\mathbf{p}, \mathbf{q}) = \sum_{b=1}^{n} p_i \ln(p_i/q_i)$.

3.4.2 A Sketch-based Approach for Feature Space Maintenance

To fulfill the bounded memory requirement of data streams Bifet et al. (2009), we maintain a fixed-size feature space over the course of the stream. We have used three versions of the algorithm. The `baselineSketch` uses the original algorithm by Metwally et al. (2005). In our setting the sketch saves pairs of (word, occurence count). To account for the importance of frequent words we use the `fadingSketch` version. The last version is the `adwinSketch` which uses the ADWIN algorithm Bifet and Gavalda (2007) as change detector to remove a saved word when it is significantly not used anymore in the stream. All different sketches allow us to maintain a fixed size feature space over the stream. Depending on the sketch type though, the update of the feature space varies. That is the sketch decides to remove not frequent words to accommodate new words coming from the stream. We devote Section 3.5.2 on the experiments to evaluate the impact of the sketch.

3.4.3 Incorporating Feature Drifts in MNB

This last section combines the previous two components, sketch and ensemble, to introduce a MNB classifier that can tackle the stated problem. We refer to this model as `temporalMNB` . As shown we present by its algorithm 3, for each arriving document d_i, firstly its words are saved in the sketch. Then assuming the $l-1$ as the current index of periods for the priors of each class c and the conditional counts of each word v. The ensemble predicts the $\hat{n}_c(lP)$ and $\hat{n}_{v,c}(lP)$, for the l-th (next) aggregation period using the weighted average of the experts' prediction (EnsemblePredict()). Then we define $P_{\mathrm{temporal}}(c)$ as the predicted class prior probability and $P_{\mathrm{temporal}}(w_i|c)$

Algorithm 3 Temporal MNB

$\text{diffTime}, t^{\text{previous}}, l \leftarrow \{0_c, 0_1, \ldots, 0_{|V|}\}$
$x_{\text{previous}}, x_{\text{period}} \leftarrow \{[0,0]_c, [0,0]_1, \ldots, [0,0]_{|V|}\}$
$V \leftarrow \{\}$
$W \leftarrow \{\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_c, \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_1, .., \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_{|V|}\}$
$N \leftarrow \{\{\{\}, \{\}\}_c, \{\{\}, \{\}\}_1, \ldots, \{\{\}, \{\}\}_{|V|}\}$

function ensembleTrain($x_v, W_v, \hat{x}_v, t$)
  $\text{diffTime}[v] \leftarrow \text{diffTime}[v] + (t - t^{\text{previous}}[v])$
  if $\text{diffTime}[v] \leq P$ **then** ▷ Aggregate for P time units
    for all $c_j \in C$ **do**
      $x_{\text{previous}}[v][c] \leftarrow x_{\text{previous}}[v][c] + x_v[c_j]$
      $N_v[c_j][l[v]] \leftarrow x_{\text{previous}}[v][c_j]$
    end for
  else ▷ New aggregation period
    for all $c_j \in C$ **do**
      $x_{\text{period}}[v][c_j] \leftarrow x_{\text{previous}}[v][c_j]$ ▷ Observe real value
      $N_v[c_j][l[v]] \leftarrow x_{\text{period}}[v][c_j]$
      for $e = 1$ to 4 **do** ▷ Update experts' weights
        $W_v^{l[v]+1}[c_j, e] \leftarrow \frac{W_v^{l[v]}[c_j,e] exp(-(x_{\text{period}}[v][c_j] - \hat{x}_v[c_j])^2/c)}{\sum_{e=1}^{4} W_v^{l[v]}[c_j,e] exp(-(x_{\text{period}}[v][c_j] - \hat{x}_v[c_j])^2/c)}$
      end for
    end for
    $k[v] \leftarrow l[v] + 1, \text{diffTime}[v] \leftarrow 0, t^{\text{previous}}[v] \leftarrow t$ ▷ Reset aggregation
    for all $c_j \in C$ **do**
      $x_{\text{previous}}[v][c_j] \leftarrow x_{\text{previous}}[v][c_j] + x_v[c_j]$
    end for
  end if
  return $W_v^{k[v]}$
end function

function ensemblePredict(X_v, W_v, c_j, t)
  $\hat{\mathbf{x}}_{\text{experts}} \leftarrow [0,0,0,0], \hat{x}_{\text{ensemble}} \leftarrow 0$
  for $e = 1$ to 4 **do** ▷ Experts predict
    $\hat{\mathbf{x}}_{\text{experts}}[e] \leftarrow \text{Expert}_e.predict(X_v[c_j], W_v[c_j], t)$
  end for
  $\hat{x}_{\text{ensemble}} \leftarrow W_v[c_j] \cdot \hat{\mathbf{x}}_{\text{experts}}$ ▷ Ensemble predicts
  return $\hat{x}_{\text{ensemble}}$
end function

function main( )
  for all $(d_i, t_i) \in D$ **do**
    $V_{d_i} \leftarrow \text{features}(d_i)$
    $V \leftarrow \text{sketch}(V_{d_i}, V)$
    for all $c_j \in C$ **do** ▷ Predict for priors and likelihoods
      $\hat{N}_c[c_j] \leftarrow \text{ensemblePredict}(N_c, W_c, c_j, t_i)$
      for all $v_{d_i} \in V_{d_i}$ **do**
        $\hat{N}_{v_{d_i}}[c_j] \leftarrow \text{ensemblePredict}(N_{v_{d_i}}, W_{v_{d_i}}, c_j, t_i)$
      end for
    end for
    ▷ Predict for document by eq.3.12
    $\hat{c}_d = \underset{c_j \in C}{argmax} P_{temporal}(c_j) \prod_i P_{temporal}(v_i|c_j)^{\text{freq}_{v_i}^d}$
    ▷ Train for priors and likelihoods
    Observe n_c for both classes at t_i
    $W_c \leftarrow \text{ensembleTrain}(n_c, W_c, \hat{N}_c, t_i)$
    for all $v_{d_i} \in V_{d_i}$ **do**
      Observe $n_{v_{d_i}}$ for both classes at t_i
      $W_{v_{d_i}} \leftarrow \text{ensembleTrain}(n_{v_{d_i}}, W_{v_{d_i}}, \hat{N}_{v_{d_i}}, t_i)$
    end for
  end for
end function

as the respective predicted likelihood for the word. The `temporalMNB` model predicts using the following equation:

$$\begin{aligned} \hat{c}_{d_i} &= \underset{c \in C}{argmax} \underset{y \in \{0,1\}}{argmax} \\ \hat{c}_{d_i} &= \underset{c \in C}{argmax} P_{\text{temporal}}(c) \prod_{v \in V_{d_i}} P_{\text{temporal}}(v|c) \end{aligned} \tag{3.12}$$

Finally, the true class of the d_i is observed. We aggregate the observed values. When a new aggregation period (lP) is reached the ensemble observes the real outcome $n_c(lP)$, $n_{v,c}(lP)$, it incurs loss and updates its weights as shown in EnsembleTrain().

3.5 Experiment Evaluation

We evaluated our approach on a real dataset from Twitter, which is related to sentiment classification (Section 3.5.1).

The role of the sketch in feature space selection is investigated in Section 3.5.2. The prediction accuracy is evaluated in Section 3.5.3, where we compare our approach to several baselines, listed below:

- `accumulativeMNB` Bifet and Frank (2010), the original MNB for streams where the word-class and class counts are accumulated over the stream. This is an incremental approach, however it does not deal with drifts Spiliopoulou et al. (2016).
- `fadingMNB` Wagner et al. (2015), adapts `accumulativeMNB` for concept drifts by introducing a fading function that decays (the accumulated over the stream) class counts and conditionals counts based on the recency of their observation. The fading function depends on the decay factor λ which controls how fast the observations age and on the aggregation period (e.g., daily vs weekly).
- `aggressivefadingMNB` Wagner et al. (2015), is a variation of `fadingMNB` that stores the decayed counts and applies ageing over them, thus leading to faster adaptation. The `aggressivefadingMNB` depends on the same parameters as `fadingMNB`.

We implemented our approach in MOA, Version 2017.06 Bifet et al. (2010). For the sketch, we used its publicly available implementation from the MOA-tweetreader package Bifet et al. (2011). For ARIMA, we used the available Java library[1]. Our implementation is available at a public repository[2].

[1] `Java` TimeSeries Library: https://github.com/signaflo/java-timeseries.
[2] GitHub Repository:https://github.com/damianosmel/temporalMNB

We employed prequential evaluation over a sliding window of 1,000 instances over each day of the dataset. As evaluation measures, we use accuracy and kappa Bifet and Frank (2010). Accuracy is the percentage of correctly predicted instances over all instances of the sliding window. However, accuracy cannot indicate if the model achieved a high value by overfitting on the current data distribution. To this end, the kappa statistic is also employed that normalizes the model performance by that of a chance classifier. We report on both average and standard deviation of those measures for each sliding window of each day.

3.5.1 Datasets

TSentiment Go et al. (2009) consists of 1.6M tweets collected over a period of three months (April 6 - June 25, 2009) annotated as positive or negative through distant supervision using emoticons as proxies for the class labels. We preprocessed the data set following the typical preprocessing steps for sentiment analysis [Jianqiang and Xiaolin (2017), Angiani et al. (2016)]. Firstly, we treated negations spliting them into two words for example we substitute "can't" with "can not". Then, we removed hashtags, mentions and URLs and converted all remaining words to lowercase. We replaced each positive emoticon with the word "EMO_POS" and each negative with the word "EMO_NEG". We substituted a word with repetitions of letters with the normal writing of the word for example "cooool" to "cool". We expanded acronyms using the dictionary of Angiani et al. (2016). We removed punctuation, numbers and whitespaces. Also we used stemming, keeping the common base form of each word. Finally, we removed stop words using the list in Manning et al. (2008).

The dataset is in overall balanced (50% positive, 50% negative tweets) but the empirical class distribution changes with time as shown in Figure 3.3 where the tweets are aggregated on a daily basis. As we can see, at the end of the monitoring period only the negative class is observed. This is an example of concept drift. No feature drifts are known for this data set.

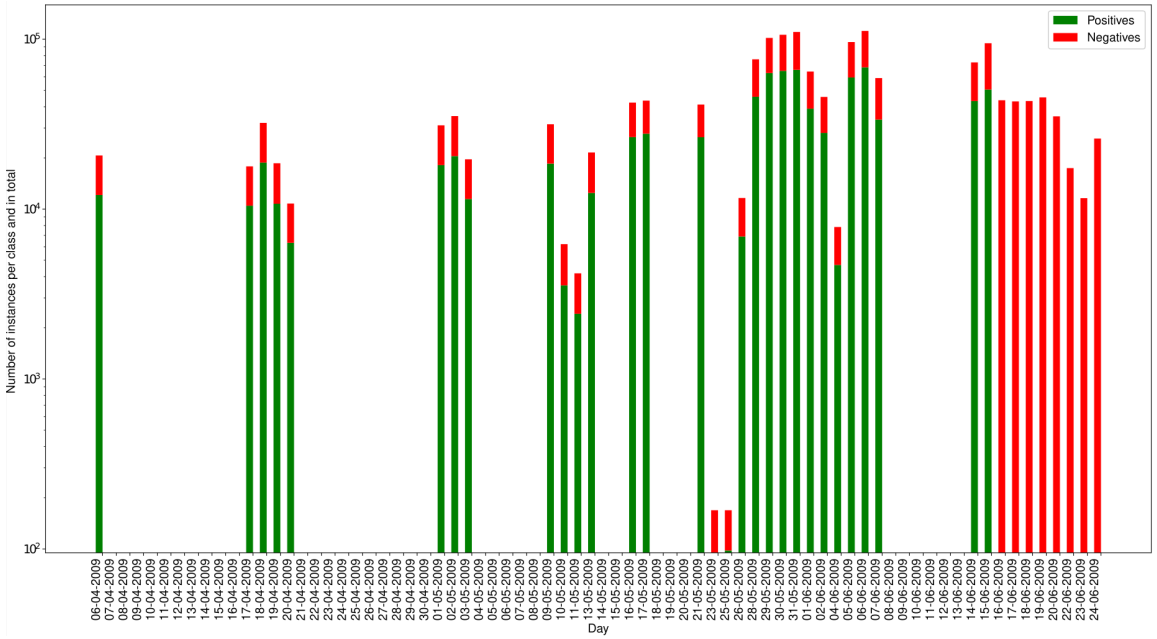


Figure 3.3. TSentiment: Class distribution over the stream (daily aggregation).

Email data Katakis et al. (2010) consists of 1,500 emails of 913 attributes. The task is to predict if an email is interesting or not for a given user. The authors know the trends of drift for some words, namely "medicine", "baseball" and "space".

Spam data Katakis et al. (2010) consists of 9,324 emails of 500 attributes. The task is to predict if an email is spam or ham. The authors use this data set as a complementary to **Email** data set to experiment with a data set where the features evolve smoother compared to the **Email** data set. For both of these data sets, we preprocessed the data set using the same techniques as before. As these two data sets are small we did not use the sketch component but we kept all found words. The goal using this data set is to verify that our proposed ensemble can capture such already known trends.

3.5.2 Sketch Evaluation

The sketch (c.f., Section 3.4.2) allows us to adapt to feature drifts in a memory-efficient way, by controlling the number of maintained words (parameter n_f). The adaptation rate of the sketch and consequently of the feature space depends on the selected sketch type (`baselineSketch`, `fadingSketch`, `adwinSketch`). In this section we evaluate the effect of n_f (sketch size) and of the different sketch types (sketch adaptation type).

Sketch Size

We experimented with three different sketch sizes n_f: 3,000, 5,000 and 10,000 distinct words. We fixed the type of sketch to `baselineSketch` that has the lowest ability to adapt to changes. We evaluate the impact of n_f on stream classification using the `accumulativeMNB` classifier.

As shown in Figure 3.4, the performance has two phases, before and after the point of the class drift. Before class drift, the larger sketch size results to the best performance, after change the smallest sketch size performs better. This is reasonable, as during stable phases the larger the sketch the more frequent words are kept thus improving the performance of the learner. However, when a change occurs, a smaller sketch will faster forget the old words and will start maintaining more recent words. A trade-off is $n_f = 5,000$, which performs decently in terms of both accuracy and memory usage. So, for the rest of the experiments we fixed n_f to 5,000 distinct words.

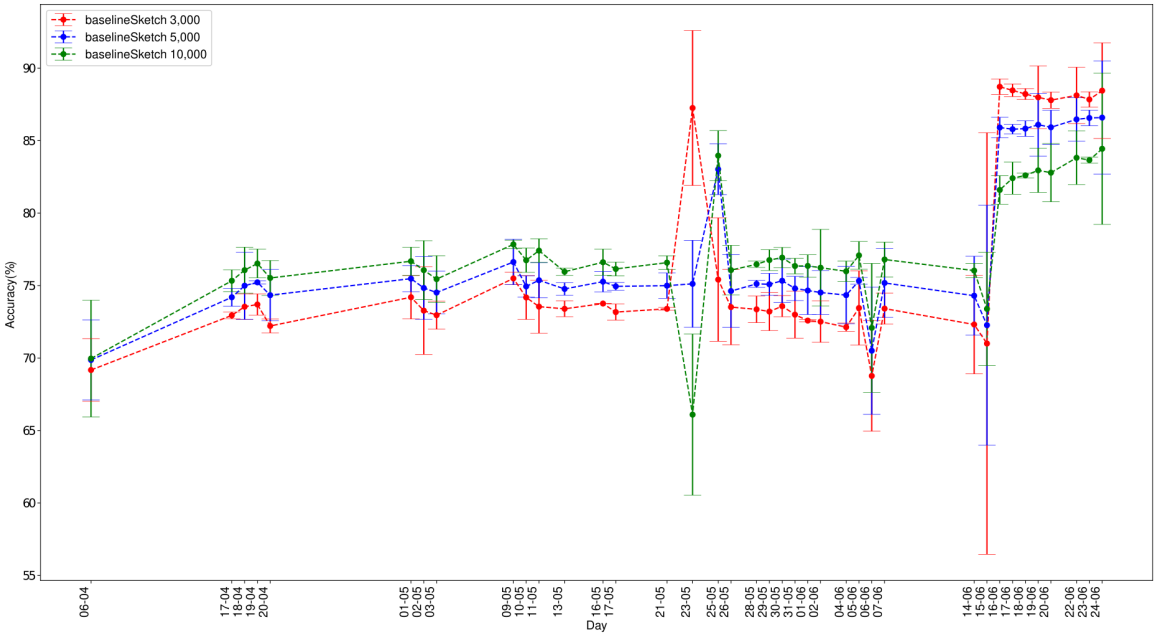


Figure 3.4. TSentiment: Effect of sketch size (daily aggregated stream, 1,000 instances evaluation window, `accumulativeMNB` classifier, `baselineSketch` adaptation type).

Sketch Adaptation Type

We evaluate the performance of the different sketch types, `baselineSketch`, `fadingSketch` and `adwinSketch`, using a fixed sketch size of $n_f = 5,000$ words. Again, we use `accumulativeMNB` as the learning model.

From the results, in Figure 3.5, we can see that the performance has again two

phases separated by the point of class drift. Before the drift, the `fadingSketch` performs better closely followed by the `baselineSketch`. After drift, `adwinSketch` performs better, followed again by `baselineSketch`. The difference in the behavior of `fadingSketch` and `adwinSketch` before and after the drift point is due to their update rate. The `adwinSketch` adapts faster to changes as it replaces words which display significant drop in their usage. On the other hand, `fadingSketch` downgrades words based on their recency without counting for changes in their usage frequency. Since we are interested in a sketch that performs decently in both times of stability and times of change, we chose `baselineSketch` that was consistently the second best performing sketch for the experiments hereafter.

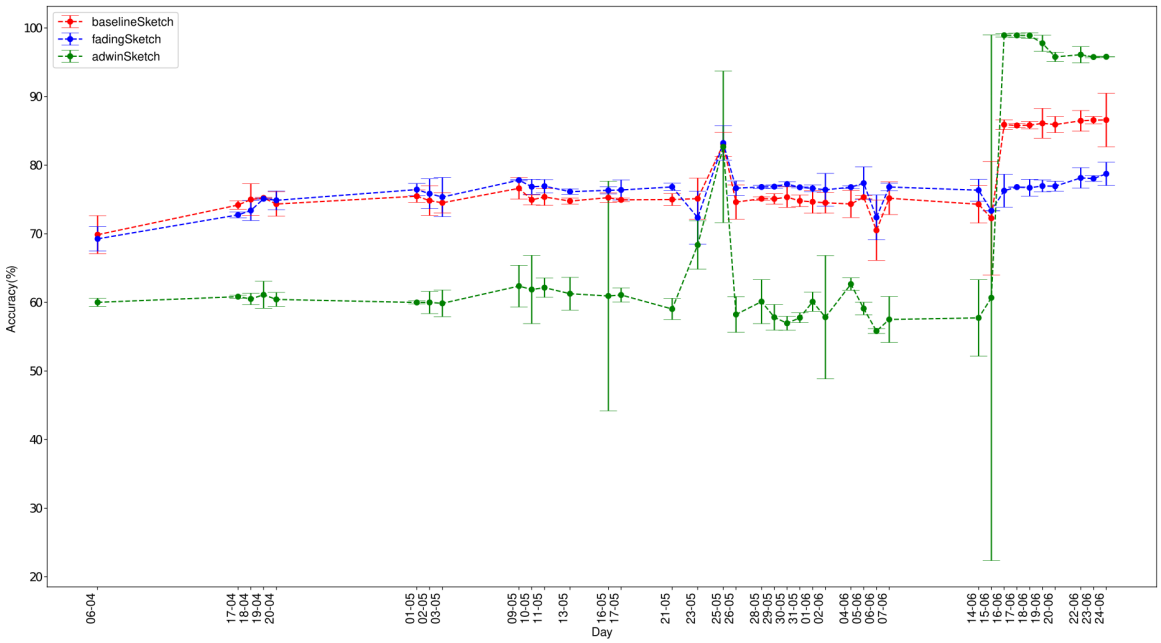


Figure 3.5. TSentiment: Effect of sketch adaptation type (daily aggregated stream, 1,000 instances evaluation window, `accumulativeMNB` classifier, $n_f = 5,000$ words).

Sketch Variability

Thus far, we have evaluated the effect of sketch size and sketch adaptation type on the performance of a stream classifier. In this section, we evaluate the sketch variability, i.e., how the words in the sketch are replaced over the stream. Recall that due to the memory constraint, only a fixed number of n_f words can be maintained over the stream.

To this end, we plot i) *the existence of change* as the percentage of instances introducing new words over the total number of instances of the day. Moreover, we are interested in the degree of such changes. To this end, we also plot ii) *the degree of change* as the percentage of new introduced words in the documents with new words

normalized with the length of the document and averaged over all documents with new words. The results are shown in Figure 3.6 (blue for (i) and red for (ii)).

We can observe that approximately half of the tweets introduce new words (blue axis) over all the course of the stream. However, the percentage of change for each such tweet is roughly 20%, i.e one new word is found in every five words of a document (red axis). We observe a large variability in the sketch in the beginning of the stream (from 6/4 to 15/4), which is probably due to the instable initialization of the sketch on 6/4. The variability scores are lower after the class drift point (Figure 3.3). A large drop is also observed on 23/5 where we observe the smallest number of documents introducing new words. However, the actual number of such documents is small as we can see from Figure 3.3.

Consequently, we can state that the `baselineSketch` produces a feature space that does not drastically change. Thus the resulted space can be employed by the learning component.

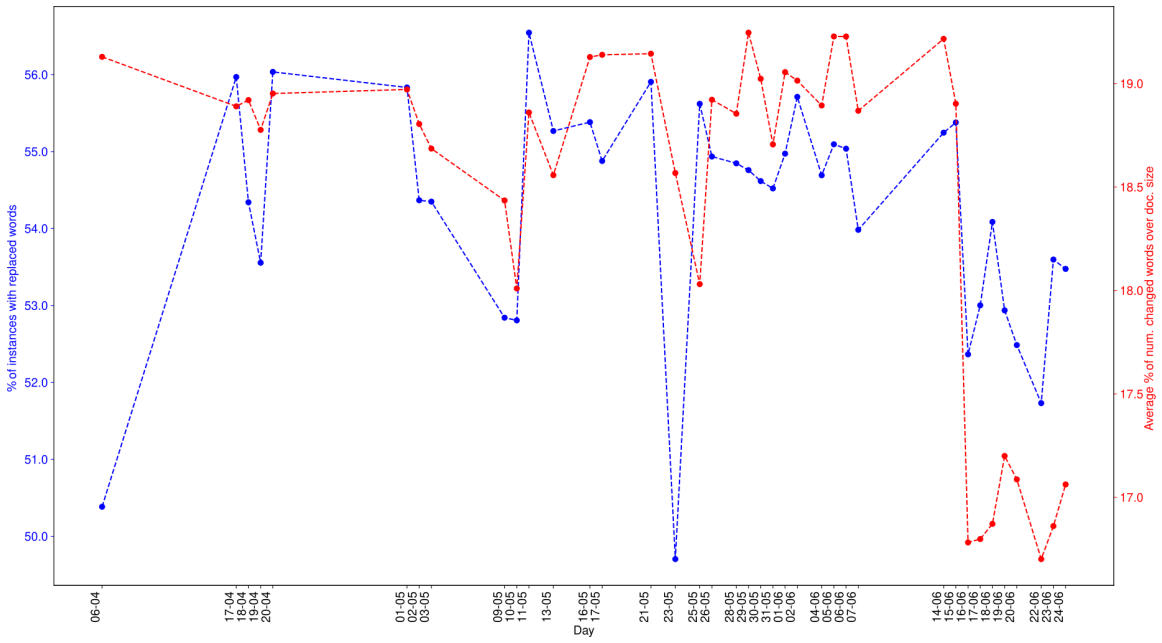


Figure 3.6. TSentiment: Variability of sketch, left y axis (blue): % of tweets introducing new words per day, right y axis (red): average % new words per document over all documents introducing new words (daily aggregated stream, `baselineSketch` with $n_f = 5,000$ words).

3.5.3 Performance Evaluation

Parameter Setting

Before comparing our approach to the baselines, we tuned their parameters. That is, for the `fadingMNB` and the `aggressivefadingMNB` we tested for $\lambda = \{0.1, 1, 2\}$ and aggregation of *second* and *hour* based on the results of Wagner et al. (2015). The best values for these two baselines is *hourly* aggregation and $\lambda = 0.1$. This means halving the conditional count after 10 hours of observing the word.

For the ensemble, we set the aggregation period P based on two facts. First, we intend to use value of P such that the ensemble can follow potential drifts of the class or the word feature. Secondly, the series for the class has very short interval as it comes by each instance but the respective series for a word has a wider interval. Thus, we tuned the parameters to capture distinct trends using aggregation period of $P = 1$ *second* for the class prior and $P = 1$ *minute* for the conditional counts of each word. For the first model, *Poisson*, we do not need to tune parameters as it is parameter-free. For the *seasonal* model, we set the sliding window history to $\gamma_s = 1$ week, i.e $H_{\text{Seasonal}} = 2$ week, capturing seasonal events during two weeks period. As the first two models capture long term dependencies we will select parameters for the last two to capture short term dependencies. Thus, for the *ARIMA* model, we use $(p = 1, d = 1, q = 1)$ to capture non-stationary time dependencies, over the last 50 periods, $H_{\text{ARIMA}} = 50P$. Following the same intuition, with the *EWMA* model that captures short time-dependencies we used a sliding window of size $\gamma_e = 22$ periods, $H_{\text{EWMA}} = 22\ P$. We set the values for fading factors of *Seasonal* and *EWMA* equal to $\alpha_s = 0.9$ and $\alpha_e = 0.1$ respectively, using the corresponding equations from Holt (2004). Experimentally, running the first three days of the stream and observing the maximum value for the class prior and the conditional count of a word we set $B = 30$, consequently $\tilde{c}_L = 1800$.

Performance

In Figure 3.7, the accuracy of the different approaches over the TSentiment dataset is depicted. We can observe that our approach performs always better than the baselines except for two days 23/05 and 25/05. After manual observing the distribution of timestamps in these two days we found that on both days we had only 169 instances having a time span of 2 minutes, on 23/05 there are instances from 18:04:32 to 18:06:34 and on 25/05 are from 10:42:48 to 10:44:22. Given our parameter selection, the ensemble samples every second for the class prior and every hour for the likelihood of each word, all changed words found in these days cannot be used for learning and consequently for their prediction the Laplace correction is used. From Figure 3.6, we can observe that the percentage of documents with changed words is close to 50% and 56% and the average number of changed words in these documents is close to 2 out of 10. Thus, in addition having words with short life span, the half of the

documents introduce new words which again span a short period of time not allowing the ensemble to be used for prediction. On the contrary, a method that the update of the model for a word is per instance, and not per aggregation period, should not show this drop in accuracy. This is true for the other two competitive baselines, the `accumulativeMNB` and `fadingMNB`.

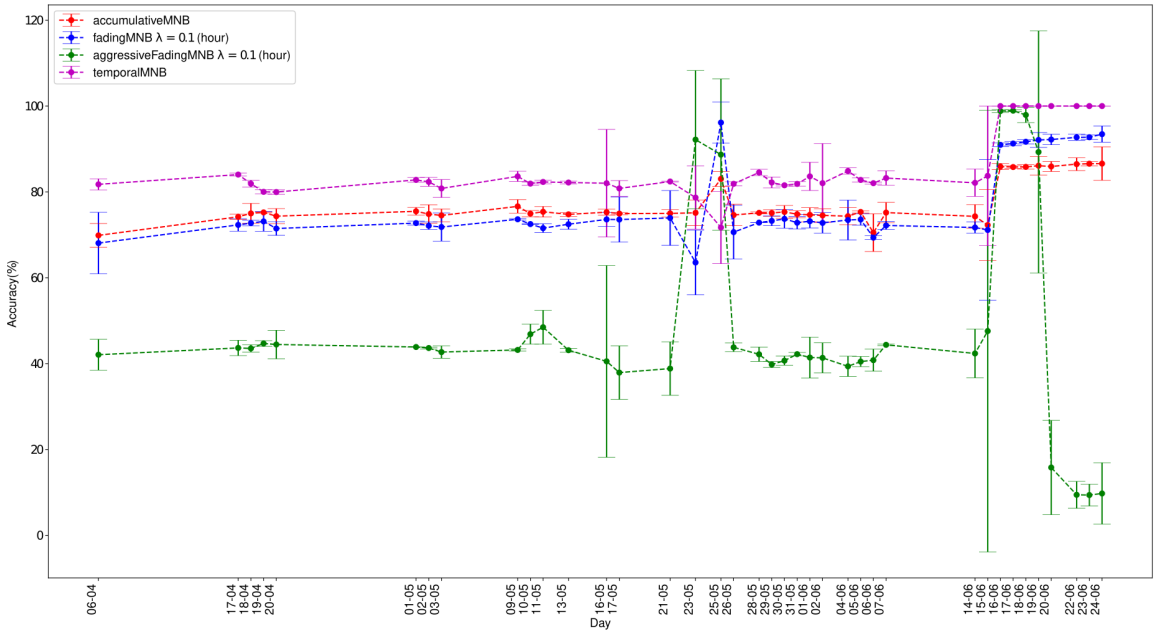


Figure 3.7. TSentiment: Accuracy over the stream for the different methods.(daily aggregated stream, 1,000 instances evaluation window, `baselineSketch`, $n_f = 5,000$ words)

We also plot the *kappa* measure in Figure 3.8 that normalizes the accuracy of the classifier with that of a chance predictor. Again, our approach outperforms the rest of the approaches over the stream, with the only exception of one day (25/05). This is caused because the time span of instances is shorter than the sampling rate of the ensemble as previously explained.

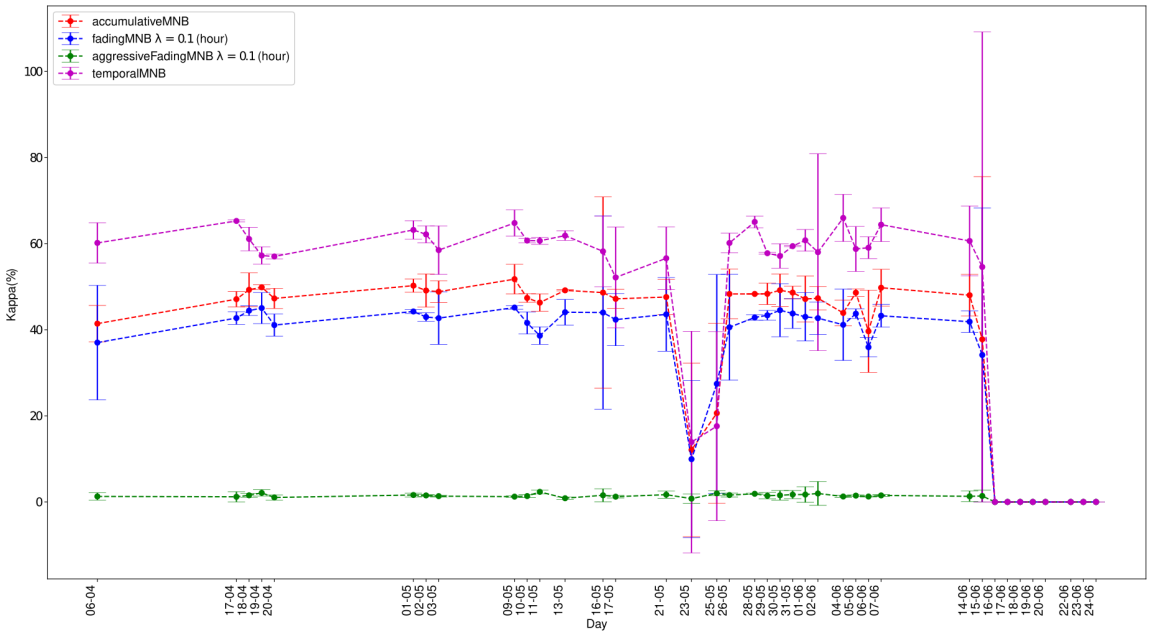


Figure 3.8. TSentiment: Kappa over the stream for the different methods.(daily aggregated stream, 1,000 instances evaluation window, `baselineSketch`, $n_f = 5,000$ words)

To check the statistical significance of our findings, we performed the McNemar's test[3]. We found that `temporalMNB` makes different predictions compared to the best baseline, `accumulativeMNB`, significantly with $p < .001$.

3.5.4 Efficiency

The running times for the different methods are presented in Table 3.1. Our approach is 713 times slower for total learning per instance comparing to the `accumulativeMNB` baseline; this is due to the employment of four different predictors for the ensemble. However, our approach can still perform at modest speed with an average of $2.53 \cdot 10^{-2}$ seconds per instance for the total learning process. Finally, the needed main memory to run this algorithm for the **TSentiment** is maximum 20 *GB*.

Method	Training	Testing	Total
`accumulativeMNB`	$4.85 \cdot 10^{-5}$ (1)	$1.67 \cdot 10^{-5}$ (1)	$6.53 \cdot 10^{-5}$ (1)
`fadingMNB`	$7.14 \cdot 10^{-6}$ (0.1)	$2.31 \cdot 10^{-5}$ (1.3)	$3.03 \cdot 10^{-5}$ (0.5)
`aggressivefadingMNB`	$9.66 \cdot 10^{-6}$ (0.2)	$2.04 \cdot 10^{-5}$ (1.2)	$3.01 \cdot 10^{-5}$ (0.5)
`temporalMNB`	$8.34 \cdot 10^{-5}$ (1.72)	$2.52 \cdot 10^{-2}$ (1,508.9)	$2.53 \cdot 10^{-2}$ (387.4)

Table 3.1. Execution times per instance in seconds and relative times compared to the `accumulativeMNB` in parenthesis.

[3]McNemar's test: https://en.wikipedia.org/wiki/McNemar's_test.

3.5.5 Ensemble Validity

Parameter Setting

Compared to the previous data set we note that this data set does not have time stamps so we assume periods over number of instances. Then, we tuned the parameters of the baselines. For the `fadingMNB` and `aggressivefadingMNB`, we found $\lambda = 2.0$ and $\lambda = 0.1$ over a period of 50 instances for the **Email** and **Spam** data sets respectively. We also tuned the parameters for `temporalMNB` . We found the same parameters for both data sets. For both the class prior and the conditional counts of words we use $P = 2$ instances. The *Poisson* model is parameter-free. For the *seasonal* model, we set the sliding window history to $\gamma_s = 2$ instance groups, i.e $H_{\text{Seasonal}} = 2$ groups, where an instance group is $25P$ instances. For the *ARIMA* model, we used $(p = 1, d = 1, q = 1)$ to capture non-stationary time dependencies, over the last 150 periods, $H_{\text{ARIMA}} = 150\ P$. For the *EWMA* model, we used a sliding window of size $\gamma_e = 22$ periods, $H_{\text{EWMA}} = 22\ P$. We set the values for fading factors of *Seasonal* and *EWMA* equal to $\alpha_s = 0.9$ and $\alpha_e = 0.1$ respectively, using the corresponding equations from Holt (2004). Experimentally, running the first instances of the stream and we observed the maximum value for the class prior and the conditional count of a word we set $B = 2$ and $B = 6$, consequently $\tilde{c}_L = 8$ and $\tilde{c}_L = 72$ for prior and the conditional count respectively.

Performance

We plot the accuracy and kappa measure every 50 instances in order to be compared to the evaluation of the original research work of Katakis et al. (2010). The accuracy plots are shown in Figure 3.9. We can observe that the ensemble achieves competitive results for both data sets. For the **Email** data set the `temporalMNB` cannot achieve the best results as it uses experts of sliding windows and this data set contains abrupt class changes. However, after an abrupt class change, for example at the batch of 350 instances, the ensemble recovers to its normal accuracy after two batches of instances. For the **Spam** data set which has a more gradual class change the `temporalMNB` achieved better performance competitive to `fadingMNB`.

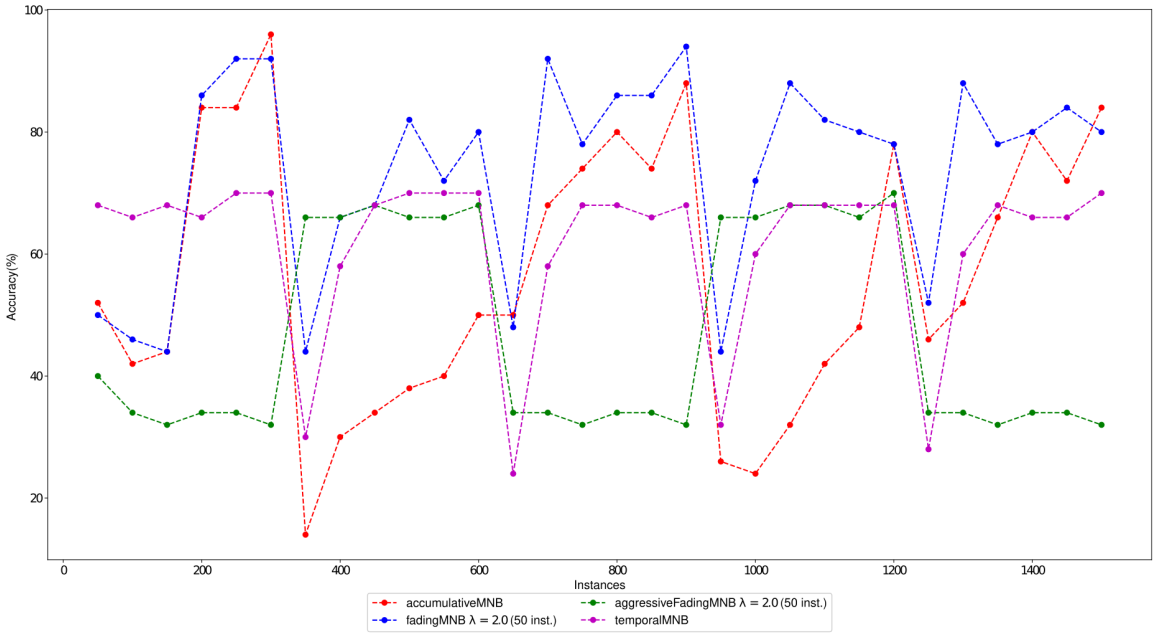


Figure 3.9. Email: Accuracy of batches of 50 instances over the stream.

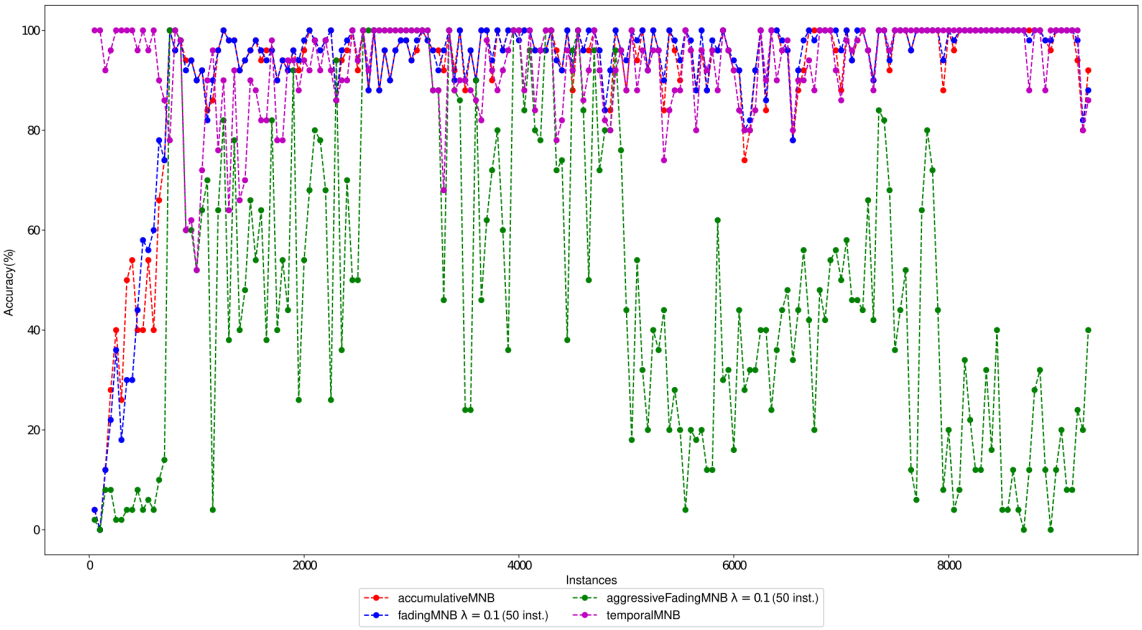
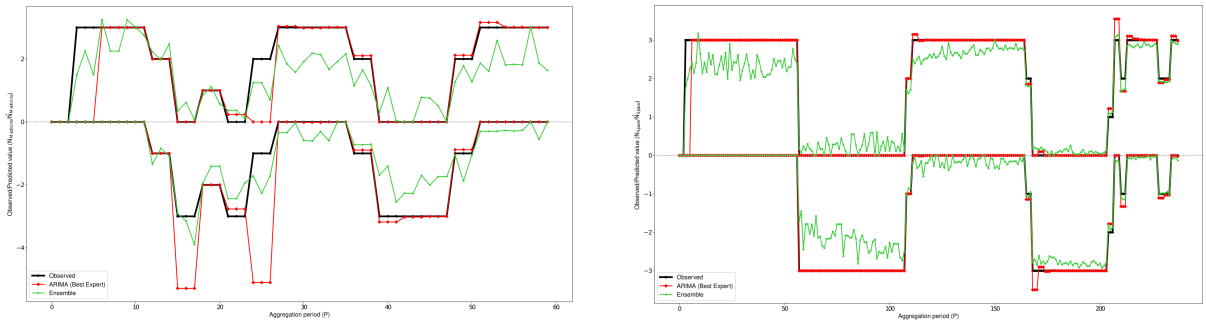


Figure 3.10. Spam: Accuracy of batches of 50 instances over the stream.

Validity

As for the **Email** data set the authors know the trends of three words, “medicine”, “space” and “baseball”, we aim to validated the first stated theorem 2. To this end, we

plotted the observed value and the predicted values of the best single expert (ARIMA) and of the ensemble for "medicine" and "space" as shown in the Figure 3.11. From this figure, we observe that truly the ensemble follows the best single expert, and the observed value, in an optimal way much lower than the given upper $c\ln M = 99.8$.



(a) "Medicine". (b) "Space".

Figure 3.11. Email: Observed and predicted conditional counts for the words "medicine" and "space" by the best single expert and the ensemble. The positive y-axis shows the counts for the positive class and the negative for the negative class.

3.6 Conclusion

To sum up, to answer our first research question, we have proposed a method to tackle concept and feature drift using two components. The first component is a sketch to maintain an updated feature space over the stream. The second is an ensemble to average out potential different trends of a feature. For the average out mechanism, we can guarantee that we can find the best single or probabilistic combination of trends representing the drift of each feature. Our experiments on textual streams demonstrated that our approach achieves competitive results compared to baselines.

4

dom2vec: Assessable domain embeddings and their use for protein prediction tasks

In this chapter we present our contributions made for the second research question RQ2, defined in 1.2.2: how we can devise protein embeddings that can be intrinsically evaluated in a quantitative manner? Can such embeddings outperform established sequence embeddings? What is the minimum amount of training data to observe *no* performance gains for the embeddings use?

Predicting biological properties of unseen proteins is shown to be improved by the use of protein sequence embeddings. However, these sequence embeddings have the caveat that biological metadata do not exist for each amino acid, in order to measure the quality of each unique learned embedding vector separately. Therefore, current sequence embedding cannot be intrinsically evaluated on the degree of their captured biological information in a quantitative manner. We address this drawback by our approach, dom2vec, by learning vector representation for protein domains and not for each amino acid base, as biological metadata do exist for each domain separately. To perform a reliable quantitative intrinsic evaluation in terms of biology knowledge, we selected the metadata related to the most distinctive biological characteristics of a domain, which are its structure, enzymatic, and molecular function.

Notably, dom2vec obtains an adequate level of performance in the intrinsic assessment—therefore, we can draw an *analogy* between the local linguistic features in natural languages and the domain structure and function information in domain architectures. Moreover, we demonstrate the dom2vec applicability on protein prediction tasks, by comparing it with state-of-the-art sequence embeddings in three downstream tasks. We show that dom2vec outperforms sequence embeddings for toxin and enzymatic function prediction and is comparable with sequence embeddings in cellular location prediction.

4.1 Introduction

A primary way in which proteins evolve is through rearrangement of their functional/structural units, known as *protein domains* Moore et al. (2008); Forslund and Sonnhammer (2012). The domains are independent folding and functional modules, and thus they exhibit conserved sequence segments. Prediction algorithms exploited this information and used, as input features, the domain composition of a protein for various tasks. For example, Chou and Cai (2002) classified the cellular location, and Forslund and Sonnhammer (2008); Doğan et al. (2016) predicted the associated Gene Ontology (GO) terms. There exist two ways to represent domains; either by the linear order in a protein, *domain architectures* Scaiewicz and Levitt (2015), or by a graph where nodes are domains and edges connect domains that co-exist in a protein Moore et al. (2008); Forslund and Sonnhammer (2012).

Moreover, Yu et al. (2019) investigated whether the domain architectures had grammar as a natural spoken language. They compared the bi-gram entropy of *domain architectures* for Pfam domains Sonnhammer et al. (1998) to the respective entropy of the English language, showing that although it was lower than the English language, it was significantly different from a language produced after shuffling the domains. Prior to this result, methods had exploited the *domain architecture* representation to various applications, such as fast homology search Terrapon et al. (2013) and retrieval of similar proteins Marchler-Bauer et al. (2016).

Word embeddings are unsupervised learning methods which have, as input, large corpora, and where they output a dense vector representation of words contained in the sentences of these documents based on the distributional semantic hypothesis, that is, the meaning of a word can be understood by its context. Thus, a word vector represents local linguistic features, such as lexical or semantical information, of the respective word. Several methods to train word embeddings have been established, for example, Collobert et al. (2011); Mikolov et al. (2013a); Pennington et al. (2014). These representations have been shown to hold several properties, such as analogy and grouping of semantically similar words Mikolov et al. (2013b); Drozd et al. (2016). Importantly, these properties are learned without the need of a labeled data set. Word embeddings are currently the mainstream input for neural networks in the Natural Language Processing (NLP) field, as firstly, they reduce the feature space, compared to 1-hot representation, and secondly, they provide word features that encapsulate relations between words based on linguistic features. The use of word embeddings improved the performance on most of the tasks, such as sentiment analysis or Named Entity Recognition (NER) Attardi et al. (2015).

Various methods used to create embeddings for proteins have been proposed Asgari and Mofrad (2015); Yang et al. (2018); Bepler and Berger (2019); Asgari et al. (2019); Heinzinger et al. (2019); Alley et al. (2019); Buchan and Jones (2020). ProtVec fragmented the protein sequence in 3-mers for all possible starting shifts, then learned embeddings for each 3-mer and represented the respective protein as the average of its

constituting 3-mer vectors Asgari and Mofrad (2015). SeqVec utilized and extended the Embeddings from Language Models (ELMo) Peters et al. (2018) to learn a dense representation per amino acid residue, resulting in matrix representations of proteins, created by concatenating their learned residue vectors Heinzinger et al. (2019).

Focusing on their word segmentation, we note that learning embeddings for each amino acid or 3-mer may not always reflect evolutionary signals. That is, a pair of proteins with low sequence similarity is still a member of the same protein superfamily, preserving similar function Loewenstein et al. (2009).

The previous embedding approaches evaluated the learned representations intrinsically, in a *qualitative* manner. They averaged out the whole protein amino acid embeddings to compute the aggregated vector. Then, known biological characteristics of proteins are used, such as biophysical, chemical, structural, enzymatic, and taxonomic, as distinct colors in a reduced 2-D embedding space. In such visualizations, previous embedding approaches reported the appearance of distinct clusters of proteins, each consisting of proteins with similar properties. For downstream evaluation, they measured the improvement of performance in downstream tasks.

Concerning the qualitative intrinsic evaluation, two caveats exist. First, researchers averaged out the protein amino acid vectors, where consequently, this qualitative evaluation is not related in a straightforward way with each learned embedding vector trained per amino-acid. In addition, this averaging-out operation may not reveal the function of the most important sites of a protein, meaning the comparative result holds a low degree of biological significance. Second, we argue that the presented qualitative evaluations lack the ability to assess different learned embeddings in a sophisticated manner. This is because there is no systematic way to *quantitatively* compare 2-D plots of reduced embedding spaces, each produced by a protein-embedding method in investigation.

Indeed for word embeddings, there has been an increase in methods to evaluate word representations intrinsically and in a quantitative manner, such as Schnabel et al. (2015); Lastra-Díaz et al. (2019). Having such evaluation metrics allows us to validate the knowledge acquired *per each word vector* and use the best-performing space for downstream tasks. However, intrinsic evaluations of current amino acid embedding representations are prevented by incomplete biological metadata at amino acid level, for all disposed proteins, in the UniProtKnowledgeBase (UniProtKB) The UniProt Consortium (2017).

To address this limitation in quantitative intrinsic evaluations of protein sequence embeddings, we present our approach with five major contributions:

1. Our `dom2vec` approach is developed, in which words are InterPro annotations and sentences are the domain architectures. Then, we use the `word2vec` method to learn the embedding vector representation for each *InterPro* annotation.

2. A quantitative intrinsic evaluation method is established based on the most significant biological information for a domain—its structure and function. First,

we evaluated the learned embedding space for domain hierarchy comparing known domain parent–children relations to cosine similarity of the parent domain. Then, we investigated the performance of a nearest neighbor classifier, $C^d_{nearest}$, to predict the secondary structure class provided by SCOPe secondary structure class Fox et al. (2013) and the Enzyme Commission (EC) primary class. Finally, we equally examined the performance of the $C^d_{nearest}$ classifier to predict the GO molecular function class for three example model organisms and one human pathogen.

3. Strikingly, we observed that $C^d_{nearest}$ reaches adequate accuracy, compared to $C^d_{nearest}$ on randomized domains vectors, for secondary structure, enzymatic function, and GO molecular function. Thus, we hypothesized an analogy between word embedding clustering by local linguistic features and protein domains clustering by domain structure and function.

4. To evaluate our embeddings extrinsically, we inputted the learned domains embeddings to simple neural networks and compared their performance with state-of-the-art protein sequence embeddings in three full-protein tasks. We surpassed both SeqVec and ProtVec for the toxin presence and enzymatic primary function prediction task, and we reported comparable results in the cellular location prediction task.

5. The pre-trained protein domain embeddings are available online at https://doi.org/10.25835/0039431, to be used by the research community.

The remainder of the paper is organized as follows: related work on protein embeddings is reviewed in Section 4. The methodology used to train and evaluate dom2vec embeddings is described in Section 4. The intrinsic and extrinsic evaluation results are presented in Section 4.3.5. In Section 4.4.7, we conclude.

4.2 Background

Current studies on protein embeddings are evaluated intrinsically and extrinsically. In extrinsic evaluation, prediction measures, like performance on a supervised learning task, are most commonly used to evaluate the quality of embeddings. For example, the ProtVec work Asgari and Mofrad (2015) evaluated their proposed embeddings extrinsically by measuring their performance in predicting protein family and disorder. SeqVec Heinzinger et al. (2019) assessed their embeddings extrinsically by measuring performance on protein-level tasks, prediction of sub-cellular localization and water solubility, and residue-level tasks, and prediction of the functional effect of single amino acid mutations. However, extrinsic evaluation methods are based on a downstream prediction task, thus not measuring the biological information captured by each learned subsequence vector separately.

Previous studies evaluated the quality of their sequence embeddings *intrinsically*, by averaging the amino acid embedding vectors per protein and then drawing t-SNE visualizations Maaten and Hinton (2008) using distinct biological labels of a protein as colors, such as taxonomy, SCOPe, and EC primary class. However, this qualitative assessment hinders the selection of the best-performing embeddings, irrespective of the downstream task, because there is not a sophisticated method to rank 2-D visualizations.

Nevertheless, in NLP, the quality of a learned word embedding space is often evaluated *intrinsically in a quantitative manner* by considering relationships among words, such as analogies. Compared to qualitative evaluation, quantitative intrinsic evaluation enables assessment of the degree of biological information captured by the embeddings. This advantage allows us to choose the best set of parameters to create the embeddings that contain the highest degree of meaningful information without choosing a specific downstream task.

From all discussed protein embeddings studies, only Buchan and Jones (2020) developed quantitative intrinsic evaluation methods. To benchmark their Pfam domain embeddings, they used the following three experiments. First, they benchmarked the performance of the nearest neighbor classifier predicting the three main GO ontologies of a Pfam using its embedding vector. Second, they assessed the Matthew's correlation coefficient Matthews (1975) between Pfam embedding and first-order Markov encodings. They also assessed the vector arithmetic to compare GO conceptual binary assignment—for example, one pair was intracellular (GO:0005622) vs. extracellular (GO:0005615).

Our approach differs from Buchan and Jones (2020) in four main points. First, we trained embeddings for all domain annotations of all proteins available in Interpro. That is, we included all available *InterPro annotations*, consisting of super-family, family, single domains, and functional sites, as "words" input to the `word2vec` method. Therefore, we used a broader set of annotations for the whole spectrum of organisms. Besides, `word2vec` was developed for sentences of natural languages, which have a moderate number of words. In order to copy with this assumption for the sentence length, we resolved overlapping and redundant annotations in order to increase the number of *InterPro annotations*, making our input more *suitable* for the `word2vec` method. Second, we benchmarked over the two `word2vec` models (`CBOW` and `SKIP`) and their parameters for each experiment of our quantitative intrinsic evaluation step, and consequently, we used our assessment to choose the best embedding space. Third, we established three unique intrinsic evaluation benchmarks for domain hierarchy, SCOPe secondary structure, and EC primary class. Lastly, our approach was also extrinsically evaluated on three downstream tasks in order to show that `dom2vec` embeddings can surpass or be comparable to state-of-the-art protein sequence embeddings.

4.3 Methods

In the following, the methodology for each part of our approach is explained. A conceptual summary is presented in Figure 4.1.

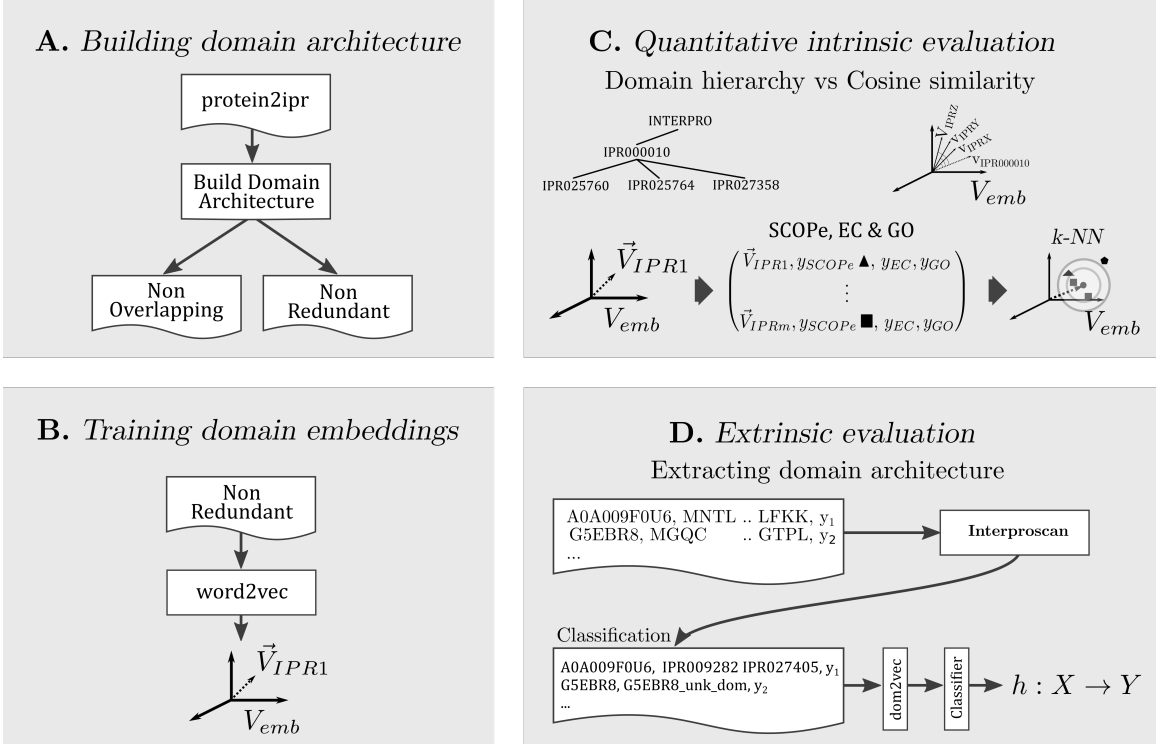


Figure 4.1. Summary of our approach divided in four parts, building two forms of domain architectures, training domain embeddings, performing intrinsic and extrinsic evaluation of *dom2vec* embeddings.

4.3.1 Building Domain Architectures

The *InterPro* database contains functional annotations for super-family, family, and single domains, as well as functional protein sites. Hereafter, we will refer to all such functional annotations as *InterPro annotations*. Furthermore, we will denote by *domain architectures* the ordered arrangement of domains in a protein. We consider two distinct strategies to represent a protein based on its domain architecture, consisting of either *non-overlapping* or *non-redundant* annotations. For both annotation types, we insert each annotation, based on the annotation's beginning and end, in an interval tree T_{hit}. For each entry of the T_{hit}, we save the annotation InterPro identifier, significance score, and length. Based on the annotation type, we apply the following two distinct strategies to create the linear domain architectures:

Non-overlapping annotations. For each overlapping region in a protein, we

keep the longest annotation out of all overlapping ones. Annotations of non-overlapping regions are included.

Non-redundant annotations. For each overlapping region in a protein, we keep all annotations with a distinct InterPro identifier. We break ties for annotations with the equal InterPro identifier by filtering in the longest one. Similarly, we keep annotations of non-overlapping regions.

For both annotation types, we sort the filtered-in annotations by their starting position. Finally, following the approach of Doğan et al. (2016), we also added the "GAP" domain to annotate more than 30 amino acid sub-sequences, which does not match any *InterPro* annotation entry.

An example of the resulting domain architectures for the *Diphthine synthase* protein is shown in Figure 4.2. All domains are overlapping, with the largest one colored in blue, and the non-overlapping annotation is the single longest domain (IPR035966). All other domains have a unique *InterPro*ID; therefore, the set of non-redundant *InterPro annotations* includes all presented domains which are sorted with respect to their starting position, and colored in green.

Applying the previous steps for all annotated proteins produces the *domain architectures*, constituting the input corpus to the following embedding module.

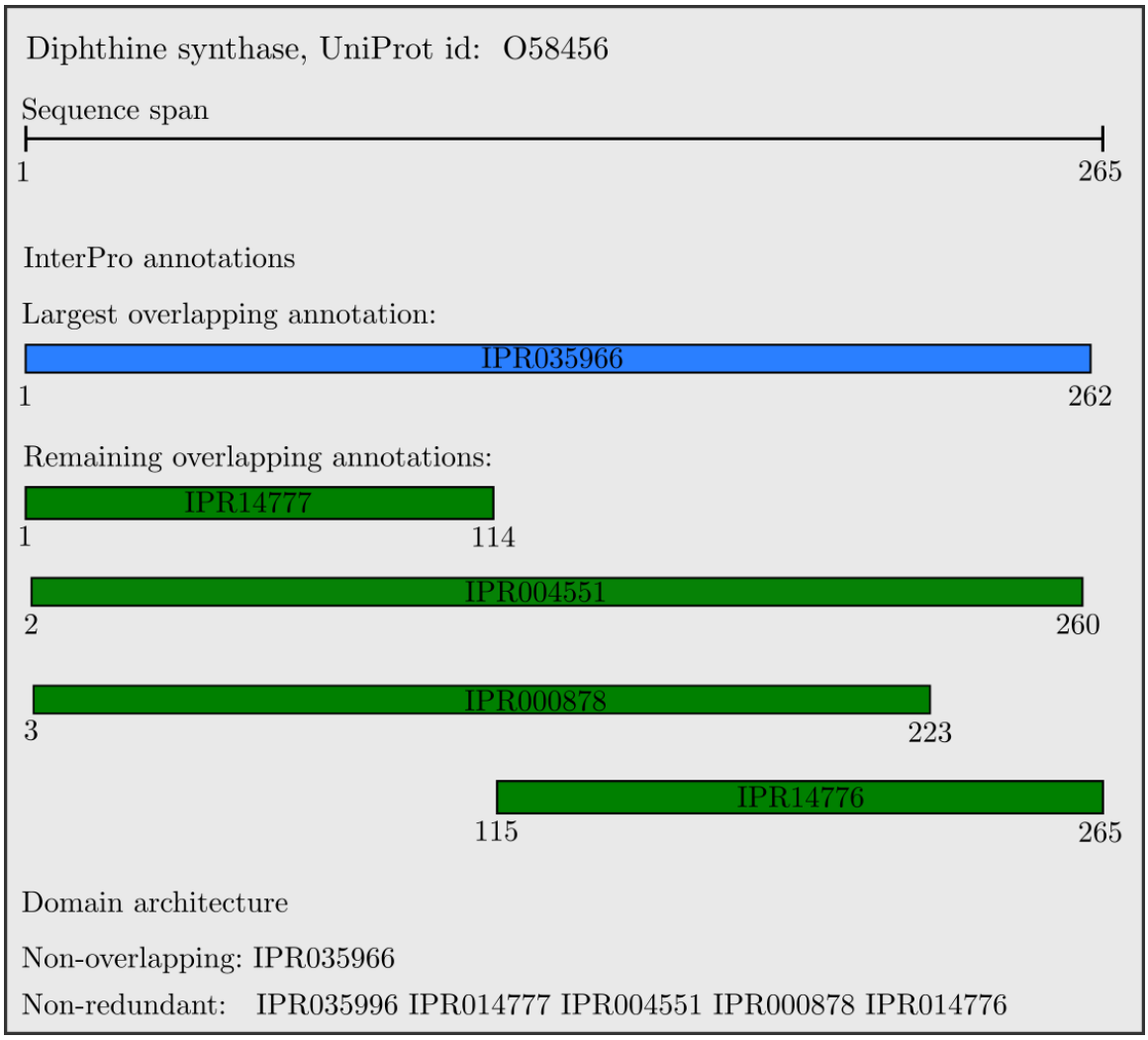


Figure 4.2. Non-overlapping and non-redundant domain architectures of the *Diphthine synthase* protein.

4.3.2 Training Domain Embeddings

Given a protein, we assumed that words were its resolved InterPro annotations and sentences were the protein domain architectures. By this assumption, we learned task-independent embeddings for each InterPro annotation using two variants of *word2vec*: a continuous bag of words and skip-gram model, hereafter denoted as `CBOW` and `SKIP` respectively. See Mikolov et al. (2013a) for technical details on the difference between these approaches. Through this training, each InterPro annotation is associated with a task-independent embedding vector.

4.3.3 Quantitative Intrinsic Evaluation

In the following, we use the metadata for the most characteristic properties of domains, in order to evaluate the learned embedding space for various hyper-parameters of `word2vec`. We propose four intrinsic evaluation approaches for domain embeddings: domain hierarchy based on the family/subfamily relation, SCOPe secondary structure class, EC primary class, and GO molecular function annotation.

We refer to the embedding space learned by *word2vec* for a particular set of hyperparameters as V_{emb}. The k nearest neighbors of a domain d is found by using the Euclidean distance, and it is denoted as $C^d_{nearest}$.

To inspect the relative performance of V_{emb} on each of the following evaluations, we *randomized* all domain vectors and ran each evaluation task. That is, we assigned to each domain vector a newly created random vector, for each unique dimensionality of embedding space, irrespective of all other embedding method parameters.

Domain hierarchy

InterPro defines a strict family–subfamily relationship among domains. This relationship is based on sequence similarity of the domain signatures. We refer to the children of domain p as S_p. We use these relationships to evaluate an embedding space, posing the following research question,

$RQ_{hierarchy}$: Did vectors of hierarchically close domains form clusters in the V_{emb}?

Evaluation We predicted the closest $|S_p|$ domains on cosine similarity of their vector to the parent vector, and we denote this predicted set as $\hat{S}_p$. For all learned embedding spaces, we measured their recall performance, $Recall_{hier}$, defined as follows:

$$Recall_{hier} = \sum_p \frac{|S_p \cap \hat{S}_p|}{|S_p|}. \tag{4.1}$$

SCOPe Secondary Structure Class

We extracted the secondary structure of Interpro domains from the SCOPe database and formed the following research question,

RQ_{SCOPe}: Did vectors of domains, with same secondary structure class, form clusters in the V_{emb}?

Evaluation We evaluated V_{emb} by retrieving $C^d_{nearest}$ of each domain. Then, we applied stratified 5-fold cross-validation and measured the performance of a k-nearest neighbor classifier to predict the structure class of each domain. The intrinsic evaluation performance metric is the average accuracy across all folds, $Accuracy_{SCOPe}$.

EC Primary Class

The enzymatic activity of each domain is given by its primary EC class Fleischmann et al. (2004) and we pose the following research question,

RQ_{EC}: Did vectors of domains, with the same enzymatic primary class, form clusters in the V_{emb}?

Evaluation We again evaluate V_{emb} using k nearest neighbors in a stratified 5-fold cross-validation setting. The average accuracy across all folds, $Accuracy_{EC}$, is again used to quantify the intrinsic quality of the embedding space.

GO Molecular Function

For our last intrinsic evaluation, we aimed to assess V_{emb} using the molecular function GO annotation. We extracted all molecular function GO annotations associated with each domain. In order to account for differences in specificity of different GO annotations, we always used the depth-1 ancestor of each annotation, that is, children of the root molecular function term, GO:0003674.

Since model organisms have the most-annotated proteins, we created GO molecular function data sets for one example of prokaryote (*Escherichia coli*, denoted *E. coli*); one example of a simple eukaryote (*Saccharomyces cerevisiae*, denoted *S.cerevisiae*); and one complex eukaryote (*Homo sapiens*, denoted *Human*). To also assess our embeddings for not highly annotated organisms, we included a molecular function data set for an example of a human pathogen (*Plasmodium falciparum*, denoted as *Malaria*). Finally, we pose the following research question,

RQ_{GO}: Did vectors of domains, with the same GO molecular function, form clusters in the V_{emb}?

Evaluation Similarly, k nearest neighbors is used here in a stratified 5-cross-validation setting. Average accuracy across all folds, $Accuracy_{GO}$, is again used to quantify performance.

4.3.4 Qualitative Evaluation

As a preliminary evaluation strategy, we used qualitative evaluation approaches adopted in an existing work. To follow the qualitative approach of ProtVec and SeqVec we also visualized the embedding space for selected domain superfamilies, to answer the following research question,

$RQ_{qualitative}$: Did vectors of each domain superfamily form a cluster in the V_{emb}?

Evaluation First, we added the vector of each domain in a randomly chosen domain superfamily to an empty space. Then, we performed principle component analysis (PCA) Pearson (1901) to reduce the space in two dimensions, and observed the formed clusters.

4.3.5 Extrinsic Evaluation

In addition, we assessed the learned V_{emb} by examining the performance change in downstream tasks. For the three supervised tasks, TargetP, Toxin, and NEW, the domain representations were used as input in simple neural networks. Next, our model performance was compared to the state-of-the-art protein embeddings, ProtVec and SeqVec.

TargetP

This data set is about predicting the cellular location of a given protein. We downloaded the TargetP data set provided by Emanuelsson et al. (2000), and we also used the non-plant data set. This data set consists of 2,738 proteins accompanied by their uniprot ID, sequence, and cellular location label, which can be nuclear, cytosol, pathway, or signal and mitochondrial. Finally, we removed all instances with a duplicate set of domains, resulting in a total of 2,418. This is a multi-class task, and its class distribution is summarized in Table 4.1 (a).

Evaluation For the TargetP, we used the mc-AuROC performance metric.

Toxin

The research workGacesa et al. (2016) introduced a data set associating protein sequence to toxic or other physiological content. We used the hard setting, which provides a uniprot ID, sequence, and the label toxin content or non-toxin content, for 15,496 proteins. Finally, we kept only the proteins with unique domain composition, resulting in 2,270 protein instances in total. This is a binary task, and the class distribution is shown in Table 4.1 (b).

Evaluation As the Toxin data set is a binary task, we used AuROC as a performance metric.

NEW

The NEW data set Li et al. (2017) contains the data for predicting the enzymatic function of proteins. For each of the 22,618 proteins, the data set provides the sequence and the EC number class. The primary enzyme class, the first digit of an EC number, is our label on this prediction task, resulting in a multi-class task. Finally, we removed all instances with duplicate domain composition, resulting in a total of 14,434 protein instances. The possible classes are six, and the class distribution is shown in Table 4.1 (c).

Evaluation The NEW data set is a multi-class task; thus, we used mc-AuROC as a performance metric.

Location	No. of proteins
Nuclear	1 072
Cytosolic	405
Pathway/Signal	605
Mitochondrial	304

(a) TargetP

Toxicity	No. of proteins
Toxin	1 747
Non toxin	523

(b) Toxin

EC Primary class	No. of proteins
Oxidoreductases	2 234
Transferases	5 232
Hydrolases	4 099
Lyases	1 124
Isomerases	731
Ligases	1 124

(c) NEW

Table 4.1. Class summary for downstream tasks. **a:** TargetP, **b:** Toxin and **c:** NEW

Data Partitioning

We divided each data set into 70/30% train and test splits. To perform model selection, we created inner three-fold cross-validation sets on the train split.

Out-of-vocabulary experiment We observed that the performance of classifiers depending on protein domains was highly dependent on the out-of-vocabulary (OOV) domains, as first discussed in Luong et al. (2015). OOV domains are all the domains contained in the test set, but not in the train. For TargetP, Toxin, and NEW, we observed that approximately 60%, 20%, and 20% of test proteins contained *at least one* OOV domain, respectively.

For the TargetP containing the highest OOV, we experimented to compensate for the high degree of OOV. We split the test set into shorter sets by an increasing degree of OOV, namely 0%, 10%, 30%, 50%, 70%, and 100%. Then, we trained models for the whole train set and benchmarked the performance on each of these test subsets.

Generalization experiment For the Toxin and NEW data sets, experiencing low OOV, we sought to investigate the generalization of the produced classifier. We increased the number of training examples that the model was allowed to learn from and we benchmarked *always* in the entire test set. To do so, we created training splits of size 10%, 20%, and 50% of the whole train set. To perform significance testing, we trained on 10 random subsamples for each training split percentage, and then tested on the separate step set. We used the paired sample t-test, the Benjamini–Hochberg multiple-test, to compare the performance between a pair of classifiers on the test set.

Simple Neural Models for Prediction

We consider a set of simple, well-established neural models to combine the InterPro annotation embeddings for each protein to perform downstream tasks, that is, for *extrinsic* evaluation tasks. In particular, we use FastText Joulin et al. (2017), convolutional neural networks (CNNs) LeCun et al. (1998), and recurrent neural networks (RNNs) with long- and short-term memory (LSTM) cells Hochreiter and Schmidhuber (1997) and bi-directional LSTMs.

4.4 Experimental Evaluation

4.4.1 Building Domain Architecture

We used the domain hits for *UniProt* proteins from *InterPro* version 75, containing 128,660,257 proteins with an *InterPro* signature, making up 80.9% of the total *UniProtKB* proteome (version 2019_06). For all these proteins, we extracted the non-overlapping and non-redundant sequences, which we process in the next section. The number of unique non-overlapping sequence was (35,183 + 1), where the added "GAP" domain and non-redundant domain was (36,872 + 1) plus the "GAP". Comparing this to the total number of domains in *InterPro* version 75, which was 36,872, we observed that non-overlapping *InterPro annotations* captured 95.42%, and the non-redundant domain captured 100% of the *InterPro* annotation entries. To enable visual comparison of the created type of domain architectures versus the downloaded InterPro annotations, in Figure 4.2 we illustrate the non-overlapping and non-redundant domain architectures of the *Diphthine synthase* protein. This same protein, *Diphthine synthase*, was picked as an example illustration for annotations in the latest InterPro work Mitchell and *et al.* (2019).

4.4.2 Training Domain Embeddings

Domain Architectures

Before applying the `word2vec` method, we examined the histograms of the number of non-overlapping and non-redundant *InterPro annotations* per protein in Figure 4.3. We observed that these distributions were long-tailed with modes equal to 1 and 3, respectively. Then, we used both CBOW and SKIP algorithms to learn domain embeddings. We used the following parameter sets. Based on the histograms, we selected the context window parameter for the word to be 2 or 5, $w = \{2, 5\}$. For the number of dimensions, we used common values from the NLP literature, $dim = \{50, 100, 200\}$. We trained the embeddings from 5 to 50 epochs with step size 5 epochs $ep = \{5, 10, 15, \ldots, 50\}$. Finally, all other parameters were set to their default values. For example, the negative sampling parameter was set to default, $ng = 5$.

4.4.3 Quantitative Intrinsic Evaluation

In the following, we evaluated each instance of learned embedding space V_{emb} for both non-overlapping and non-redundant representations of domain architectures. An instance of V_{emb} space is the embedding space learned for a combination of the product $annotation\_type \cdot w \cdot dim \cdot ep$. Consequently, the total number of embedding space instances is $|annotation\_type| \cdot |w| \cdot |dim| \cdot |ep| = 2 \cdot 2 \cdot 3 \cdot 10 = 120$. Let V^i_{emb}

denote such an embedding space instance. In the following subsection, we evaluated each V^i_{emb} for domain hierarchy, secondary structure, enzymatic primary class, and GO molecular function. Finally, all reported performances are shown *for the best-performing epoch value (ep)*.

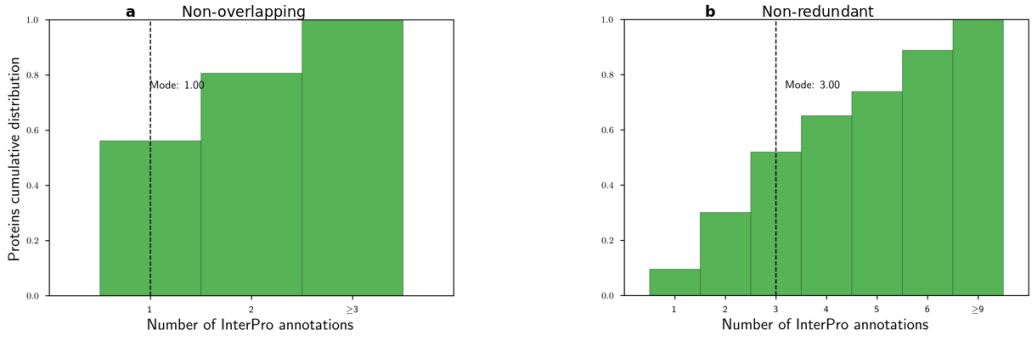


Figure 4.3. Histograms of number of *InterPro annotations* per protein. (**a**) Non-overlapping and (**b**) non-redundant *InterPro annotations*.

$RQ_{hierarchy}$: Did vectors of hierarchically close domains form clusters in the V_{emb}?

For the first research question, we loaded the parent–child tree T_{hier}, provided by *InterPro*, consisting of 2,430 parent domains. Then, for each V^i_{emb}, we compared the actual and predicted children of each parent, and we averaged out the recall for all parents.

From Tables 4.2, we observed that SKIP performed better overall, and the embeddings learned from non-redundant *InterPro annotations* always had better average recall values compared to the non-overlapping ones. The best-performing V^i_{emb} achieved average $Recall_{hier}$ of 0.538. We compared this moderate performance of V_{emb} with the performance of the randomized spaces, which was equal to 0. We concluded that our embedding spaces greatly outperformed each randomized space for domain hierarchy relation. Therefore, we admitted that the *majority* of domains of the same hierarchy were placed in close proximity in the embedding space.

(a) $Recall_{hier}$ for non-overlapping *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
`CBOW`,w=2	0.242 (*ep*=15)	0.259 (*ep*=20)	0.263 (*ep*=15)
`CBOW`,w=5	0.242 (*ep*=45)	0.252 (*ep*=30)	0.25(*ep*=15)
`SKIP`,w=2	0.287 (*ep*=20)	**0.316** (*ep*=30)	0.32 (*ep*=20)
`SKIP`,w=5	0.284 (*ep*=20)	0.302 (*ep*=30)	0.311 (*ep*=30)
random	0	0	0

(b) $Recall_{hier}$ for non-redundant *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
CBOW, w=2	0.406 (*ep*=10)	0.412 (*ep*=10)	0.414 (*ep*=5)
CBOW, w=5	0.405 (*ep*=30)	0.402 (*ep*=35)	0.382 (*ep*=10)
SKIP, w=2	0.512 (*ep*=5)	0.53 (*ep*=5)	**0.538** (*ep*=5)
SKIP, w=5	0.507 (*ep*=5)	0.525 (*ep*=5)	0.524 (*ep*=5)
random	0	0	0

Table 4.2. Intrinsic evaluation performance for domains hierarchy. **a:** Average $Recall_{hier}$ for non-overlapping *InterPro annotations* **b:** Average $Recall_{hier}$ for non-redundant *InterPro annotations*. For all sub-tables, results shown for the best performing *ep* value; if k not shown then k=2. Best performance is shown in bold case.

The histogram of average recall for best performing embedding space is shown at Figure 4.4 (a). We observe that the embeddings space brought close domains with *unknown* family-subfamily relation for almost the one third of the parent domains (827 out of 2 430).

To diagnose the reason for this moderate performance, we plotted the histogram of the number of children for each parent having recall 0, Figure 4.4 (b). We observed that most of these parents had only one child. Consequently, the embedding space should have been very homogeneous, for each of these parent child relation, in order to acquire better recall than 0.

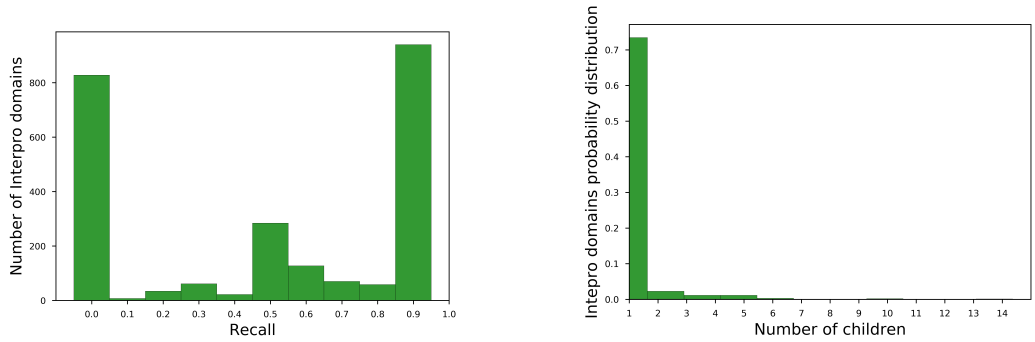


Figure 4.4. Diagnostic plots for domain hierarchy assessment task. (**a**) $Recall_{hier}$ histogram for SKIP, w=2, dim=200, ep=5 for non-redundant *InterPro annotations*and (**b**) histogram of number of children for parents with $Recall_{hier}$=0.

RQ_{SCOPe}: Did vectors of domains with the same secondary structure class form clusters in the V_{emb}?

We extracted the SCOPe class for each *InterPro* domain. This resulted in 25,196 domains with an unknown secondary structure class, 9,411 with a single secondary structure class, and 2,265 domains with more than one assigned class (multi-label). For clarity, we removed all multi-label and unknown instances, resulting in 9,411 single-labeled instances. The class distribution of the resulting data set is shown in Table 4.3.

We measured the performance of the $C^d_{nearest}$ classifier in each V^i_{emb} to examine the homogeneity of the space with respect to the SCOPe class. We split the 9,411 domains in 5-fold stratified cross-validation sets. To test the change in prediction accuracy for an increasing number of neighbors, we used different sets of neighbors, namely, $k = \{2, 5, 20, 40\}$. We summarized the results for the best-performing $C^d_{nearest}$, which was $k = 2$ for non-redundant InterPro annotations in Table 4.3c. We show the respective table for non-overlapping *InterPro annotations* in Table 4.3b. We compared these accuracy measurements to the respective ones of the random spaces, and we found that the lowest accuracy values, achieved for (non-overlapping, CBOW, $w = 5$, $dim = 200$, $ep = 15$) are twice as high as the accuracy values of the random spaces for all possible dimensions.

Consequently, we concluded that domain embeddings of the same secondary structure class formed distinct clusters in the learned embedding space.

(a) SCOPe classes

SCOPe class	No. of domains
a	1 868
b	1 806
a|b	2 303
a+b	2 320
multi-domain	304
membrane/cell	309
small	501

(b) $Accuracy_{SCOPe}$ for non-overlappping *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
`CBOW`,w=2	50.01(ep=30)	50.69(ep=25)	50.45(ep=20)
`CBOW`,w=5	49.59(ep=25)	50.03(ep=25)	48.82(ep=15)
`SKIP`,w=2	**51.83**(ep=30)	51.79(ep=20)	51.78(ep=15)
`SKIP`,w=5	51.54(ep=35)	51.65(ep=15)	51.34(ep=15)
random	22.75 (k=40)	24.18 (k=40)	23.39 (k=40)

(c) $Accuracy_{SCOPe}$ for non-redundant *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
CBOW, w=2	77.09 (ep=5)	76.35 (ep=5)	75.77 (ep=5)
CBOW, w=5	78.15 (ep=5)	76.94 (ep=5)	76.84 (ep=5)
SKIP, w=2	84.42 (ep=45)	84.42 (ep=40)	84.08 (ep=30)
SKIP, w=5	**84.56** (ep=25)	84.06 (ep=45)	83.72(ep=10)
random	23.39 (k=40)	23.49 (k=40)	22.76 (k=20)

Table 4.3. SCOPe evaluation. **a:** SCOPe class summary, **b-c:** $C^d_{nearest}$ average accuracy over all folds, $Accuracy_{SCOPe}$, for (b) non-overlapping and (c) non-redundant *InterPro annotations*. Default k=2, best performance value shown in bold case.

RQ_{EC}: Did vectors of domains, with the same enzymatic primary class, form clusters in the V_{emb}?

We processed the EC primary class, resulting in 29,354 domains with unknown EC, 7,248 domains with only one EC, and 721 with more than one EC. As before, we removed all multi-label and unknown instances, leaving 7,428 domains with known EC. We augmented a domain instance with its vector representation for each V^i_{emb}, and then we used $C^d_{nearest}$to predict the EC label. See Table 4.4 for the class distribution of the EC task.

We reported the average $Accuracy_{EC}$ obtained in embedding spaces learned using non-overlapping and non-redundant *InterPro annotations* in Table 4.4. We compared these accuracy measurements to the respective ones of the random spaces. We found that the minimum average $Accuracy_{EC}$ value was equal to 60.51 and was achieved

using (non-overlapping, CBOW, $w = 5$, $dim = 200$, $ep = 15$), presented in Subtable 4.4b. That value was approximately twice as large as the accuracy values of the random spaces for all possible dimensions; the maximum average $Accuracy_{EC}$ for random space with $dim = 100$ was 32.64.

Hence, we were able to accept that domain embeddings of the same EC primary class formed distinct clusters in a learned embedding space.

(a) EC classes

EC primary class	No. of domains
Oxidoreductases	1 102
Transferases	2 490
Hydrolases	2 190
Lyases	524
Isomerases	318
Ligases	448
Translocases	176

(b) $Accuracy_{EC}$ for non-overlapping *InterPro annotations*

Model╲ Dimension	dim=50	dim=100	dim=200
CBOW,w=2	61.23(ep=10)	61.33(ep=10)	60.66(ep=15)
CBOW,w=5	61.22(ep=20)	60.51(ep=10)	60.61(ep=15)
SKIP,w=2	63.56(ep=10)	**63.92**(ep=20)	62.58(ep=20)
SKIP,w=5	62.47(ep=10)	63.44(ep=10)	62.94(ep=15)
random	31.51 (k=40)	32.64 (k=40)	31.68 (k=20)

(c) $Accuracy_{EC}$ for non-redundant *InterPro annotations*

Model╲ Dimension	dim=50	dim=100	dim=200
CBOW, w=2	76.88(ep=5)	75.85(ep=5)	75.39(ep=5)
CBOW, w=5	80.89(ep=5)	79.89(ep=5)	77.16(ep=5)
SKIP, w=2	89.47(ep=35)	89.06(ep=40)	88.86(ep=5)
SKIP, w=5	**90.85**(ep=30)	90.41(ep=15)	90.2(ep=5)
random	33.62 (k=40)	32.06 (k=40)	32.28 (k=40)

Table 4.4. EC evaluation. **a:** EC class summary, **b-c:** average $C^d_{nearest}$ accuracy over all folds, $Accuracy_{EC}$, for (b) non-overlapping and (c) non-redundant *InterPro annotations*. Default value of neighbors k=2, best accuracy shown in bold case.

RQ_{GO}: Did vectors of domains with the same GO molecular function form clusters in the V_{emb}?

We parsed the GO annotation file of *InterPro* to extract first-level GO molecular function for domains for the four organisms. We followed the same methodology to examine the homogeneity of a V_{emb} with respect to GO molecular function annota-

tions. For each V^i_{emb}, we augmented each domain by its vector and its GO label, and we classified each domain using $C^d_{nearest}$. As before, we used 5-fold stratified cross-validation for evaluation. In our experiments, we varied the number of neighbors $k = \{2, 5, 20, 40\}$ to test its influence on the change of performance. For space limitations, we summarized the performances showing only the best average accuracy over the number of neighbors.

For *Malaria*, the best average accuracy was 76.86 (non-redundant, SKIP, w = 5, dim = 100, ep = 40) and the minimum was 56.94 (non-overlapping, CBOW, w = 5, dim = 100, ep = 10), presented in Table 4.5 (b-c) respectively. We compared this moderate minimum accuracy to the maximum level of accuracy obtained by the randomized embedding space, which was 47.57 for dim = 200.

Therefore, we concluded that `dom2vec` embeddings outperformed the random baseline by at least 10 percent.

(a) GO classes

GO class	No. of domains
Catalytic activity	676
Binding	440
Structural molecule activity	171
Transporter activity	63
Molecular function regulator	22
Transcription regulator activity	13
Cargo receptor activity	1
Molecular carrier activity	1

(b) $Accuracy_{GO}$ for non-overlapping *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
CBOW, w=2	58.2 (k=5,*ep*=35)	58.24(k=5,*ep*=5)	57.87(*ep*=5)
CBOW, w=5	57.87(*ep*=10)	56.94(*ep*=10)	57.1(*ep*=5)
SKIP, w=2	59.48(k=5,*ep*=10)	59.82(*ep*=15)	58.68(*ep*=10)
SKIP, w=5	**60.61**(*ep*=10)	59.01(*ep*=10)	59.39(*ep* = 5)
random	46.21(k=40)	46.62(k=40)	45.99(k=40)

(c) $Accuracy_{GO}$ for non-redundant *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
CBOW, w=2	67.33(*ep*=5)	64.88(*ep*=5)	61.13(*ep*=5)
CBOW, w=5	66.74(*ep*=5)	66.75(*ep*=5)	63.89(*ep*=5)
SKIP, w=2	75.79(*ep*=35)	75.64(*ep*=45)	74.91(*ep*=5)
SKIP, w=5	76.79(*ep*=10)	**76.86**(*ep*=40)	72.75(*ep*=20)
random	46.58(k=40)	46.22(k=40)	47.57(k=40)

Table 4.5. *Malaria* GO molecular function evaluation. **a:** GO class summary, **b-c:** Average $C^d_{nearest}$ accuracy over all folds, $Accuracy_{GO}$, for non-overlapping and non-redundant *InterPro annotations*, whenever k is not shown k=2, best shown in bold case.

For *E. coli*, the best accuracy score was 81.72 (non-redundant, SKIP, $w = 5$, *dim* = 50, *ep* = 5), and the minimum was 67.34 (non-overlapping, CBOW, $w = 2$, *dim* = 200, *ep* = 5), shown in Table 4.6 (b-c) respectively. Compared with the random baseline, achieving a best accuracy score of 64.46, we observed that, again, *dom2vec* was able to surpass the random baseline.

(a) GO classes

GO class	No. of domains
Catalytic activity	1 565
Binding	476
Transporter activity	211
Structural molecule activity	117
Transcription regulator activity	39
Molecular function regulator	15
Molecular carrier activity	3
Translation regulator activity	1
Molecular transducer activity	1

(b) $Accuracy_{GO}$ for non-overlapping *InterPro annotations*

Model\ Dimension	$dim = 50$	$dim = 100$	$dim = 200$
`CBOW`, w=2	67.66(k=5,ep=30)	67.46(k=20,ep=5)	67.34(k=20,ep=5)
`CBOW`, w=5	67.78(k=5,ep=5)	67.46(k=20,ep=5)	67.34(k=20,ep=5)
`SKIP`, w=2	68.15(k=5,ep=5)	67.54(k=5,ep=5)	67.75(k=20,ep=5)
`SKIP`, w=5	**69.1**(k=5,ep=5)	67.82(k=5,ep=5)	68.15(k=5,ep=5)
random	64.46(k=40)	64.46(k=40)	64.46(k=40)

(c) $Accuracy_{GO}$ for non-redundant *InterPro annotations*

Model\ Dimension	dim=50	dim=100	dim=200
`CBOW`, w=2	71.41(k=5,ep=5)	68.45(k=5,ep=5)	67.87(ep=5)
`CBOW`, w=5	74.95(ep=5)	71.69(ep=5)	68.91(ep=5)
`SKIP`, w=2	81.27(ep=5)	80.32(ep=5)	80.36(ep=5)
`SKIP`, w=5	**81.72**(ep=5)	81.64(ep=5)	80.77(ep=5)
random	64.38(k=40)	64.46(k=40)	64.38(k=40)

Table 4.6. *E.coli* GO molecular function evaluation. **a:** GO class summary, **b-c:** Average $C^d_{nearest}$ accuracy over folds, $Accuracy_{GO}$, for non-overlapping and non-redundant *InterPro annotations*. Whenever k is not shown k=2, best shown in bold case.

For *Yeast*, the best accuracy score was 75.10 (non-redundant, SKIP, $w = 5$, $dim = 50$, $ep = 50$), and the minimum accuracy value was 59.82 (non-overlapping, CBOW, $w = 5$, $dim = 50$, $ep = 50$), presented in Table 4.7 (b,c) respectively. We contrasted this to the maximum accuracy level obtained in a random space, which was 53.73 (achieved for $dim = 100$), to report that *dom2vec* vectors in $V^{E.coli}_{emb}$ captured GO molecular function classes at a much higher degree than randomized vectors.

(a) GO classes

GO class	No. of domains
Catalytic activity	1 177
Binding	585
Structural molecule activity	208
Transporter activity	112
Transcription regulator activity	46
Molecular function regulator	40
Translation regulator activity	2
Molecular transducer activity	2
Molecular carrier activity	1
Cargo adaptor activity	1

(b) $Accuracy_{GO}$ for non-overlapping *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
`CBOW`, w=2	60.05(k=20,ep=5)	59.87(k=20,ep=5)	59.87(k=20,ep=5)
`CBOW`, w=5	59.82(k=20,ep=15)	60.24(k=20,ep=5)	60.70(k=20,ep=5)
`SKIP`, w=2	60.74(k=5,ep=10)	60.79(k=5,ep=10)	61.53(k=5,ep=5)
`SKIP`, w=5	**61.38**(k=5,ep=10)	60.75(k=20,ep=5)	60.61(k=20,ep=10)
random	53.36(k=40)	53.64 (k=40)	53.64 (k=40)

(c) $Accuracy_{GO}$ for non-redundant *InterPro annotations*

Model╲ Dimension	*dim*=50	*dim*=100	*dim*=200
`CBOW`, w=2	64.37(k=5,ep=5)	64.87(k=5,ep=5)	62.4(k=5,ep=5)
`CBOW`, w=5	67.17(k=5,ep=50)	65.11(k=5,ep=5)	63.31(k=5,ep=5)
`SKIP`, w=2	73.36(k=5,ep=20)	73.86(k=5,ep=5)	72.29(k=5,ep=5)
`SKIP`, w=5	**75.1**(k=5,ep=50)	74.1(k=5,ep=10)	73.02(k=5,ep=5)
random	53.59(k=40)	53.73(k=40)	53.18(k=40)

Table 4.7. *S.cerevisiae* GO molecular function evaluation. **a:** GO class summary, **b-c:** Average $C^d_{nearest}$ accuracy over folds,$Accuracy_{GO}$, for non-overlapping and non-redundant *InterPro annotations*, best shown in bold

For *Human*, the best average performance for non-redundant *InterPro annotations* are shown in Table 4.8. The best average accuracy level was 75.96, scored by 2-NN for V^{human}_{emb} (non-redundant, SKIP, w = 5, dim = 50, ep = 40). The minimum accuracy value was 57.7, obtained by (non-overlapping, CBOW, w = 2, dim = 50, ep = 10) shown in Table 4.8b. The best performance of a random space was 37.36 (Table 4.8b). We compared the minimum accuracy level of trained spaces with the best of the random spaces. We found that the minimum accuracy achieved in the `dom2vec` spaces was 20 percentage values higher than the best performance of the random space.

(a) GO classes

GO class	No. of domains
Catalytic activity	1 945
Binding	1 583
Transporter activity	377
Molecular transducer activity	355
Structural molecule activity	262
Transcription regulator activity	203
Molecular function regulator	168
Cargo receptor activity	9
Molecular carrier activity	1
Cargo adaptor activity	1

(b) $Accuracy_{GO}$ for non-overlapping *InterPro annotations*

Model\ Dimensions	dim=50	dim=100	dim=200
`CBOW`, w=2	57.7(ep=10)	58.82(ep=10)	58.04(k=5,ep=10)
`CBOW`, w=5	58.1(ep=30)	58.9(ep=35)	58.2(ep=10)
`SKIP`, w=2	60.51(k=5,ep=15)	60.67(ep=10)	59.3(ep=10)
`SKIP`, w=5	**60.59**(k=5,ep= 35)	60.18(k=5,ep=10)	59.84(k=5,ep=10)
random	36.72 (k=40)	36.44 (k=40)	37.36 (k=40)

(c) $Accuracy_{GO}$ for non-redundant *InterPro annotations*

Model\ Dimension	dim=50	dim=100	dim=200
CBOW, w=2	66.94(ep=5)	66.32(ep=5)	66.32(ep=5)
CBOW, w=5	67.77(ep=5)	65.87(ep=5)	65.77(ep=5)
SKIP, w=2	74.77(ep=40)	74.18(ep=5)	73.14(ep=5)
SKIP, w=5	**75.96**(ep=40)	75.53(ep=10)	74.98(ep=5)
random	37.05 (k=40)	37.03 (k=20)	37.05 (k=40)

Table 4.8. *Human* GO molecular function evaluation: **a:** GO class summary, **b-c:** Average $C^d_{nearest}$ accuracy over folds, $Accuracy_{GO}$, for (b) non-overlapping and (c) non-redundant *InterPro annotations*, when k is not shown k=2, best shown in bold case.

For all four example organisms, we observed that the SKIP on non-redundant *InterPro annotations* produced V_{emb}, in which $C^d_{nearest}$ achieved the best average accuracy. For three out of the four organisms, the best performances were achieved for the lowest number of dimensions (dim = 50). In all cases, we found that the worst-performing `dom2vec` embeddings outperformed the random baselines. By these findings, we affirmed that domain embeddings of the same GO molecular function class formed distinct clusters in the learned embedding space.

4.4.4 Concluding on Quantitative Intrinsic Evaluation

Based on the previous four experiments, we aimed to evaluate the learned V_{emb} spaces and select the best domain embedding space for downstream tasks. In all experiments, the non-redundant *InterPro annotations* created better-performing embedding spaces compared to non-overlapping annotations. We reached this finding by comparing the modes of a number of annotations per protein for the two annotation types, Figure 4.3. We hypothesized that, by the very low mode for non-overlapping annotations, a mode equal to one annotation, the `word2vec` method could not produce embeddings for even the stringent context window value of two. In contrast, 52% of proteins contained less than or equal to three non-redundant InterPro annotations.

This makes `SKIP` able to produce embedding spaces by attaining the best intrinsic performance. From the individual results, we saw that the configuration of parameters (non-redundant, SKIP, $w = 5$, $dim = 50$) brought the best results in $C^{d}_{nearest}$ performance for SCOPe, EC, and GO for *E. coli*, *Yeast*, *Human*, second best for *Malaria*, and the sixth best recall (0.507) for the domain hierarchy relation. Therefore, we will denote as $V_{emb}^{best\ intrinsic}$, the space produced by (non-redundant, SKIP, $w = 5$, $dim = 50$, $ep = 50$).

4.4.5 Qualitative Evaluation

$RQ_{qualitative}$: Did vectors of each domain superfamily form a cluster in the V_{emb}?

To explore the V_{emb} in terms of the last research question, $RQ_{qualitative}$, we randomly selected five InterPro domain superfamilies to perform the visualization experiment. The selected domain superfamilies were *PMP-22/EMP/MP20/Claudin superfamily* with parent InterPro id IPR004031, *small GTPase superfamily* with parent InterPro id IPR006689, *Kinase-pyrophosphorylase* with parent *InterPro* id IPR005177, *Exonuclease, RNase T/DNA polymerase III* with parent InterPro id IPR013520, and *SH2 domain* with parent InterPro id IPR000980.

We loaded the parent–child tree T_{hier}, provided by InterPro, and for each domain superfamily starting from the parent domain, we included recursively all domains that had a subfamily relationship with this parent domain. For example, the *Kinase-pyrophosphorylase* domain superfamily had domain parent IPR005177, which in turn had two immediate domain subfamilies IPR026530 and IPR026565. The IPR026565 domain contained a subfamily domain with ID IPR017409, where consequently, the set of domains for *Kinase-pyrophosphorylase* domain superfamily was {IPR005177, IPR026530, IPR026565, and IPR017409}. We retrieved the vectors for each domain in each superfamily in the $V_{emb}^{best\ intrinsic}$. Finally, we applied principal component analysis (PCA) to produce a two-dimensional space.

Visualization of the reduced space is depicted in Figure 4.5. Domain embeddings

of each superfamily are organized in well-separated clusters. The cluster of the *Exonuclease, RNase T/DNA polymerase III* superfamily had the highest dispersion of all presented superfamilies. By this finding, we could answer the research question with the following: Embedding vectors of the same superfamily are well-clustered in the trained V_{emb}.

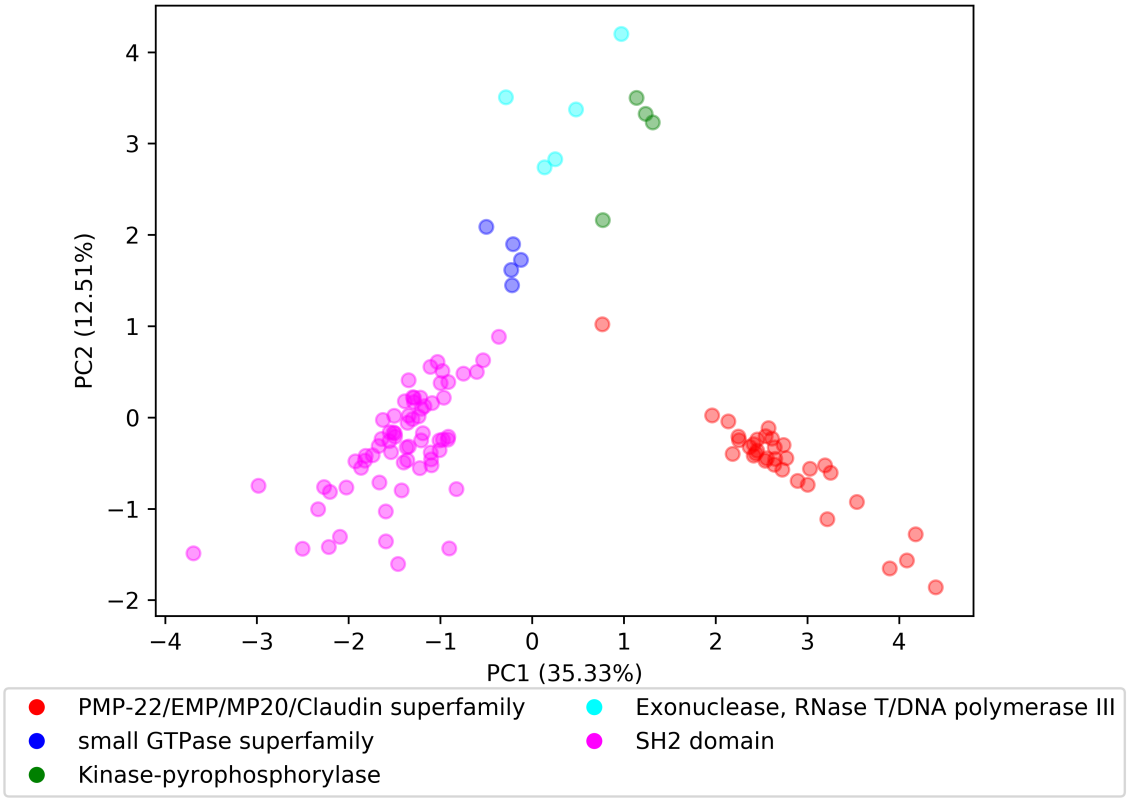


Figure 4.5. Domain vectors for five domain superfamilies in the `dom2vec` space.

4.4.6 Extrinsic Evaluation

Extracting Domain Architectures

For each data set that contained the *UniProt* identifier for the protein instance, we extracted the domain architectures for non-redundant *InterPro annotations*, already created in Section "Building domain architectures". For all proteins whose *UniProt* identifier could not be matched, or for data sets not providing the protein identifier, we used *InterProScan* Jones and *et al.* (2014) to find the domain hits per protein. For proteins without a domain hit after *InterProScan*, we created a protein-specific, artificial protein-long domain; for example, we assigned to the protein G5EBR8, a protein-long domain named "G5EBR8_unk_dom".

Model Selection

To select which simple neural model we should compare to the baselines, we performed hyperparameter selection using an inner, three-fold cross-validation on the training set; the test set was not used to select hyperparameters. We used common parameters, with a dropout of 0.5, batch size of 64, an Adam optimizer Kingma and Ba (2015) with learning rate of 0.000, weight decay for the last fully connected layer of 0, and number of epochs equal to 300. As a final hyperparameter, we allowed updates to the learned domain embeddings, initialized by selected `dom2vec` embeddings. The results are shown in Table 4.9.

Model ╲ Data set	TargetP	Toxin	NEW
CNN, size=(1,2),filters=200	0.9191	0.9074	0.9845
CNN, size=1,filters=128	**0.9288**	0.8957	0.9844
FastText (uni-gram)	0.8829	0.9029	0.9818
LSTM, dim=512,layer=1	0.9103	0.9025	0.9857
bi-LSTM, dim=512,layer=1	0.921	0.9052	0.9857
SeqVecNet dim=32	0.9017	0.9086	**0.9876**
SeqVecNet dim=512	0.9206	**0.9145**	0.9864
SeqVecNet dim=1024	**0.9228**	0.9034	0.9861

Table 4.9. Average performance of simple neural architectures using as input `dom2vec` on inner three-fold cross validation. For Toxin AuROC is shown and for the two other data sets mc-AuROC is shown. Best values shown in bold case.

4.4.7 Running Baselines

Then, we used the same network as the one in the right side of Figure 5 of Heinzinger et al. (2019); we refer to this network as SeqVecNet. Namely, the network first averages the 100 (ProtVec) or 1024 (SeqVec) dimensional embedding vector for a protein; it then applies a fully connected layer to compress a batch of such vectors into 32 dimensions. Next, a ReLU activation function (with 0.25 dropout) was applied to that vector, followed by batch normalization. Finally, another fully connected layer was followed by the prediction layer. As the third baseline, we added the *1-hot* of domains in order to investigate the performance change compared to `dom2vec` learned embeddings.

Evaluation

For TargetP, we sought to investigate the effect of OOV on the produced classifier compared to sequence-based embeddings classifiers which do not experience OOV, as their used sequence features were highly common in both the train and test sets. For the Toxin and NEW datasets, we benchmarked the generalization of the produced

classifier compared to the sequence-based embeddings classifiers. Finally, for both kinds of experiments, we used the trained models on each test set. Hence, this evaluation shows how differences in the training set affect performance on the test set. The resulting performances are shown in Figure 4.6.

Out-of-vocabulary experiment For TargetP, we validated that OOV will affect the performance of domains dependent classifiers. That is, for OOV in the range of 0–30%, the `dom2vec` classifier was comparable to the best-performing model, SeqVec. However, when OOV increased even further, then the performance of our model dropped, though still being competitive with the SeqVec. `dom2vec` greatly outperformed the *1-hot* representation, validating the NLP assumption that unsupervised embeddings improve classification on unseen words—in this context, protein domains—compared to *1-hot* word (domain) vectors.

Generalization experiment For both Toxin and NEW, *dom2vec* significantly outperformed SeqVec, ProtVec, domains *1-hot* vectors, and Benjamini–Hochberg multiple-test corrected p-value < 0.05. In the Toxin data set, we observed that ProtVec learned the less variant model, but with the trade-off obtaining the lowest performance (mc-AuROC). For the NEW data set, the *dom2vec 1-hot* representation was the second-best representation outperforming SeqVec and ProtVec, allowing us to validate the finding that domain composition is the most important feature for enzymatic function prediction, as concluded by Li et al. (2017).

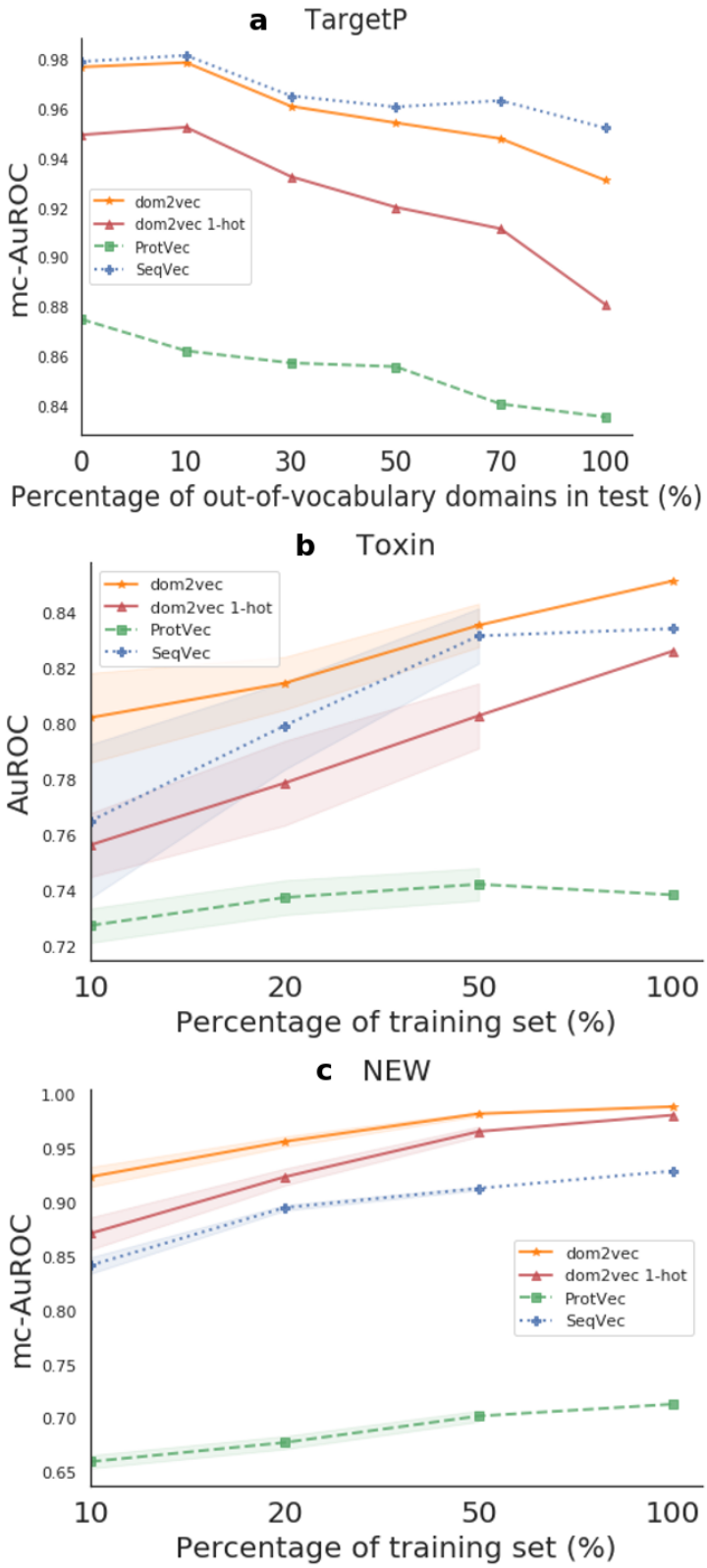


Figure 4.6. Downstream performance. Subfigure (**a**) refers to the OOV experiment, while (**b**) and (**c**) refer to the generalization experiment. The marked points represent the mean performance on the test set, and the shaded regions show one standard deviation above and below the mean.

4.5 Conclusion

To answer the second research question, we presented `dom2vec`, an approach for learning quantitatively assessable protein domain embeddings using the word2vec method on *domain architectures* from *InterPro annotations*.

We have shown that `dom2vec` adequately captured the domain SCOPe structural information, EC enzymatic function, and the GO molecular function of each domain with such available metadata information. However, `dom2vec` produced moderate results in the domain hierarchy evaluation task. After investigating the properties of domain families that `dom2vec` produces these moderate results, we concluded that `dom2vec` cannot capture the domain hierarchy, mostly for domain families of low cardinality. We argue that by using more complex classifiers compared to $C^{d}_{nearest}$, we could gain in hierarchy performance, but this was not the scope of our evaluation.

Importantly, we did discover that `dom2vec` embeddings captured the most distinctive biological characteristics of domains, secondary structure, and enzymatic and molecular function for an individual domain. That is, `word2vec` produced domain embeddings which clustered *sufficiently* well by their structure and function class. Therefore, our finding supported the accepted modular evolution of proteins Moore et al. (2008), in a data-driven way. It also made possible a striking analogy between words in natural language that clustered together in *word2vec* space Mikolov et al. (2013b), and protein domains in domain architectures that clustered together in `dom2vec` space. **Therefore, we parallel the semantic and lexical similarity of words to the functional and structural resemblance of protein domains.** This analogy may augment the research on understanding the nature of rules underlying the domain architecture grammar Yu et al. (2019). We are confident that this interpretability aspect of `dom2vec` will allow researchers to apply it reliably, so as to predict biological features of novel domain architectures and proteins with identifiable InterPro annotations.

In downstream task evaluation, `dom2vec` significantly outperformed domain *1-hot* vectors and state-of-the-art sequence-based embeddings for the Toxin and NEW data sets. For the TargetP, `dom2vec` was comparable to the best-performing sequence-based embedding, Seqvec, for OOV up to 30%. Therefore, we recommend using `dom2vec` in combination with sequence embeddings to boost prediction performance.

5

GenOtoScope: Towards automating ACMG/AMP classification of variants associated with congenital hearing loss

In this chapter we introduce our bioinformatics software tool, `GenOtoScope`. `GenOtoScope` automates half of the evidence-based criteria for the classification of variant pathogenicity based on the ACMG/AMP guidelines. Thus, this work contributes on the research questions of third research question RQ3, defined in 1.2.3. Namely, how can we classify genomic variants associated with a polygenic heretidary disease, like hearing loss, in a time-efficient and standardized manner? and how the automated classification can be easily usable by diagnostics personel with limited bioinformatics experience?

Since next-generation sequencing (NGS) has become widely available, large gene panels containing up to several hundred genes can be sequenced cost-efficiently. However, the interpretation of the often large numbers of sequence variants detected when using NGS is laborious, prone to errors and are often difficult to compare across laboratories. To overcome this challenge, the American College of Medical Genetics and Genomics and the Association for Molecular Pathology (ACMG/AMP) have introduced standards and guidelines for the interpretation of sequencing variants. Additionally, disease-specific refinements have been developed that include accurate thresholds for many criteria, enabling highly automated processing. This is of particular interest for common but heterogeneous disorders such as hearing impairment. With more than 200 genes associated with hearing disorders, the manual inspection of possible causative variants is particularily difficult and time consuming.

To this end, we developed the open-source bioinformatics tool `GenOtoScope`, which automates the analysis of all ACMG/AMP criteria that can be assessed without further individual patient information or human curator investigation, including the refined loss of function criterion ("PVS1"). Two types of interfaces are provided: (i) a command line application to classify sequence variants in batches for a set of patients

and (ii) a user-friendly website to classify single variants.

We compared the performance of our tool with two other variant classification tools using two hearing loss data sets, which were manually annotated either by the ClinGen Hearing Loss Gene Curation Expert Panel or the diagnostics unit of our human genetics department. `GenOtoScope` achieved the best average accuracy and precision for both data sets. Compared to the second-best tool, `GenOtoScope` improved accuracy metrics by 25.75% and 4.57% and precision metrics of 52.11% and 12.13% on the two data sets, respectively. The web interface is freely accessible. The two interfaces, along with all source code, documentation and example outputs can be accessed via the GenOtoScope web page: http://genotoscope.mh-hannover.de:5000/.

5.1 Introduction

Due to the establishment of modern high-throughput next generation sequencing (NGS) technologies, an ever-increasing amount of sequencing data can be generated. Nevertheless, a whole exome sequencing (WES) file contains approximately 60,000 variants per proband. Consequently, laboratories have to overcome the hurdle of processing this vast amount of data to link the genotype to phenotype Linder et al. (2021). Notably, the manual classification of variants, by expert curators, is not only time-consuming, but even more, prone to inconsistent functional interpretation and pathogenicity classification of a variant between distinct laboratories Berrios et al. (2021).

To address this challenge, the American College of Medical Genetics and the Association for Molecular Pathology (ACMG/AMP) published a set of evidence-based criteria to classify patients variants in five classes of pathogenicity, "benign" (class 1), "likely benign" (class 2), variants of uncertain significance ("VUS") (class 3), "likely pathogenic" (class 4), and "pathogenic" (class 5) Richards et al. (2015). According to these guidelines, various information about a variant of interest and its associated phenotype (e.g., population data, computational data, functional data, segregation data) can be assorted into 28 well-defined categories that function as evidence criteria for a variant to be pathogenic or benign. The acronym of each criterion is a composite of P (pathogenic) or B (benign) and the respective graded strength level, A (stand-alone), VS (very strong), S (strong), M (moderate), P (supporting), followed by a numerical identifier denoting different types of information. The graded combination of evidence criteria results in the five-tier classification system mentioned above. An overview of the evidence-based criteria is depicted in Fig. 5.1 and the classification scheme shown in Fig. 5.2.

To specialise for a diverse set of phenotypes with distinct penetrance, allelic and genetic heterogeneity, ACMG updated its classification criteria for specific hereditary diseases, for example hereditary (breast/ovarian) cancer Lee et al. (2019) or cardiomyopathy Kelly et al. (2018), through the ClinGen Variant Curation Expert Panels

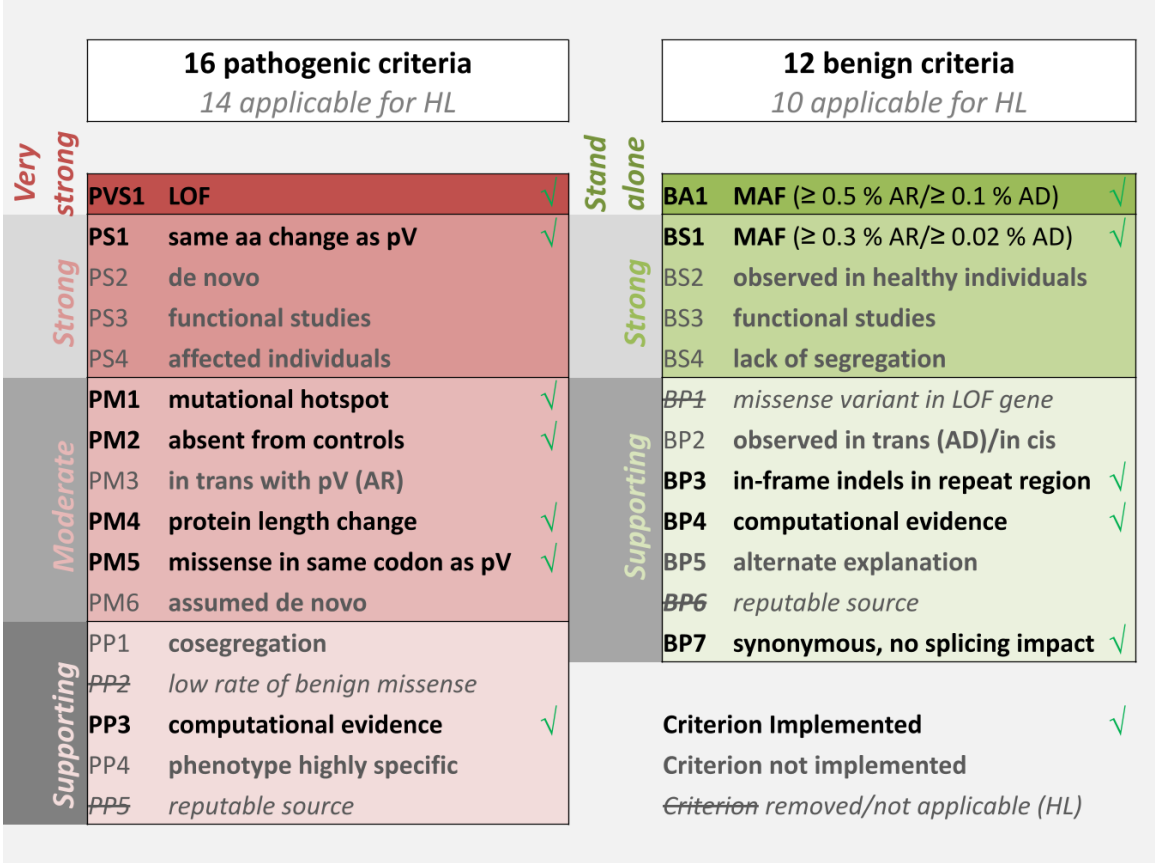


Figure 5.1. Overview of ACMG/AMP evidence-based criteria. Green mark shows implemented criteria. Grey font shows not implemented criteria. Striked-through shows removed or not applicable criterion for hearing loss. Thresholds shown for BA1 and BS1 are specific for HL.

(VCEP). Even more than the general ACMG recommendations, these disease-specific criteria are predestined for a computerized approach due to their precise thresholds.

Hearing loss (HL) is the most common sensory disorder with a high impact on the quality of social and work life of the patient. A genetic etiology can be linked to approximately 50% of the affected individuals Shearer et al. (2017). Besides various forms of nonsyndromic hearing loss (NSHL) affecting only the function of the ear, HL can also be a symptom of a superordinate disorder involving other organ systems (syndromic hearing loss). Thus, HL is very heterogeneous with well over 100 genes known to be associated with monogenetic NSHL and more than 400 distinctive syndromes comprising HL as one of their characteristic symptoms as well Shearer et al. (2017). Because of this tremendous versatility combined with the high overall frequency of occurrence, we selected HL as a model disease for the development of our software tool. Therefore, the algorithms presented in this work are by default set to HL-specific thresholds, but can be easily modified to suit other medical conditions.

Pathogenic	**(i)** 1 Very strong (PVS1) **AND** **(a)** ≥1 Strong (PS1–PS4) **OR** **(b)** ≥2 Moderate (PM1–PM6) **OR** **(c)** 1 Moderate (PM1–PM6) and 1 supporting (PP1–PP5) **OR** **(d)** ≥2 Supporting (PP1–PP5) **(ii)** ≥2 Strong (PS1–PS4) **OR** **(iii)** 1 Strong (PS1–PS4) **AND** **(a)** ≥3 Moderate (PM1–PM6) **OR** **(b)** 2 Moderate (PM1–PM6) and ≥2 Supporting (PP1–PP5) **OR** **(c)** 1 Moderate (PM1–PM6) and ≥4 supporting (PP1–PP5)
Likely Pathogenic	**(i)** 1 Very strong (PVS1) and 1 moderate (PM1–PM6) **OR** **(ii)** 1 Very strong (PVS1) and 1 Supporting (PM2_Supporting) (AR, HL) **OR** **(ii)** 1 Strong (PS1–PS4) and 1–2 moderate (PM1–PM6) **OR** **(iv)** 1 Strong (PS1–PS4) and ≥2 supporting (PP1–PP5) **OR** **(v)** ≥3 Moderate (PM1–PM6) **OR** **(vi)** 2 Moderate (PM1–PM6) and ≥2 supporting (PP1–PP5) **OR** **(vii)** 1 Moderate (PM1–PM6) and ≥4 supporting (PP1–PP5) ($P_{pathogenicity}$ = 0.9, relaxed $P_{pathogenicity}$ = 0.49988)
Benign	**(i)** 1 Stand-alone (BA1) **OR** **(ii)** ≥2 Strong (BS1–BS4)
Likely Benign	**(i)** 1 Strong (BS1–BS4) and 1 supporting (BP1–BP7) **OR** **(ii)** 1 Strong (BS1) (HL) **OR** **(iii)** ≥2 Supporting (BP1–BP7) ($P_{pathogenicity}$ = 0.025, relaxed $P_{pathogenicity}$ = 0.05072)
Uncertain significance	**(i)** Other criteria shown above are not met **OR** **(ii)** the criteria for benign and pathogenic are contradictory

Figure 5.2. ACMG/AMP classification scheme evidence-based criteria. The table contains 2 columns. The right column contains sufficient conditions of triggered criteria that result to the left column, pathogenicity class. Sufficient combination of criteria specified for HL are marked with (HL). Pathogenicity probability and its relaxed version are shown for the criteria combinations with the lowest strength that can result to "likely benign" or "likely pathogenic" class.

There are orthogonal approaches to perform the challenging classification of HL variants. The first approach is to use machine learning models that predict the pathogenicity of variants with respect to hearing loss phenotype such as DVPred Bu et al. (2022). However, such approaches do not output the triggered ACMG evidence-based criteria supporting the classification result, thus the classification results cannot be easily interpreted by the human variant curator. An alternative is to gather and organise all known and classified variants for HL, into an open-access database. Gene4HL Huang et al. (2021), is a database that the human curator can import, export and find a HL variant of interest. Nevertheless, the database approach has limited validity in the variant assessment process, assuming that the human curator aims to assess the pathogenicity of a variant, which is not already catalogued in the database. The last approach is to automate ACMG/AMP guidelines specified for HL. In this work, we focus on such an approach. The main reasons are that such a bioinformatics tool will annotate and classify all variants of WES experiment with respect to HL phenotype and output the triggered ACMG evidence-based criteria supporting each pathogenicity classification. Consequently, this tool provides to the human curator, pathogenicity classification for a variant not yet classified and reports the triggered ACMG/AMP criteria allowing for an interpretable classification.

Oza et al. (2018) work has introduced disease-specific evidence-based ACMG/AMP criteria, to facilitate the challenging classification of variants for HL. Noteworthy, 2 criteria were marked as not applicable (PP2 and BP1) for HL and 2 criteria (PP5 and BP6) have been sorted out as generally not applicable. Application of the remaining 24 adjusted criteria has been shown to achieve better classification performance compared to the standard evidence-based criteria for known HL-related variants Patel et al. (2021). A recently published bioinformatics tool, VIP-HL Peng et al. (2020), automates 13 out of the 24 evidence-based criteria specified for HL. However, VIP-HL is an online tool that accepts only a single variant per time, thus hindering the automatic and time-efficient interpretation of all variants of WES files for a set of investigated patients, for a heterogeneous condition as HL.

To address this limitation of VIP-HL, we present `GenOtoScope`, a bioinformatic tool which accepts as input a genomic variant file (VCF) and compute the pathogenicity class and pathogenicity probability for each input variant, based on Oza et al. (2018) and Tavtigian et al. (2018). To this end, we designed and implemented algorithms to automate all the evidence-based criteria that need no further individual patient information or human curator investigation. This results to 12 implemented criteria, out of the 24 criteria in total, namely PVS1 (all strengths), PS1, PM1, PM2 (PM2 supporting), PM4, PM5 (PM5 strong), PP3, BA1, BS1 (BS1 supporting), BP3, BP4 and BP7. We will provide `GenOtoScope` as an open-source project, accessible as command line application to classify the WES patients files and as an online tool to classify a single genomic variant of interest.

We benchmarked the performance of `GenOtoScope` compared to two established classification tools, InterVar Li and Wang (2017) and VIP-HL, in two HL data sets. These data sets consist of manually curated HL variants. `GenOtoScope` outperformed the other two classification algorithms, both, in terms of accuracy and precision. Finally, we investigated the reasons for this best performance of `GenOtoScope`, by calculating the difference between the activation frequencies of a tool over the manual curation, for each evidence-based criterion.

In summary, our contributions are:

- introduce `GenOtoScope` in two forms, a command line application for bioinformatics experts to classify WES VCF files of a set of patients and a web-based application for non-bioinformatics experts to classify single variants.
- compare `GenOtoScope` classification performance to InterVar and VIP-HL for two manual annotated HL data sets.
- make `GenOtoScope` an open-source bioinformatics tool, therefore enabling the research community to extend the tool for other diseases.

5.2 Methods

5.2.1 Automating the examination of ACMG evidence criteria

`GenOtoScope` currently implements 12 out of 24 ACMG evidence-based criteria specified for hearing loss Oza et al. (2018). More specifically these criteria are PVS1 (all strengths), PS1, PM1, PM2 (PM2 supporting), PM4, PM5 (PM5 strong), PP3, BA1, BS1 (BS1 supporting), BP3, BP4 and BP7. Based on class category the implemented criteria are sorted in 7 pathogenic and 5 benign criteria. With respect to the data types needed for ACMG criteria, we categorize our implemented criteria into 3 population data criteria, 8 computational and predictive data criteria and 1 functional data criterion. The comparison of `GenOtoScope` with VIP-HL and InterVar is summarised in Table 5.1.

Tool	Implemented Criteria	Phenotype-specific	Open Code Implementation	Open Annotation Data Sets	Command-line Application (variant sets)	Web Application (single variant)	Evaluation Data Sets
InterVar	18/28 Benign: 8/12 Pathogenic: 10/16	No	Yes	No	Yes	Yes	• *De novo* variants in neurodevelopmental disorders (9,305)* • Benign & pathogenic ClinVar (49,167)* • Pathogenic HGMD (616)* • All CLINVITAE (11,696)*
VIP-HL	13/24 Benign: 6/10 Pathogenic: 7/14	Yes	No	Yes	No	Yes	• Pilot VCEP-HL (50)** • All deafness-related ClinVar (4,948)*
`GenOtoScope`	12/24 Benign: 5/10 Pathogenic: 7/14	Yes	Yes	Yes	Yes	Yes	• All VCEP-HL (158)** • Manually classified by diagnostics unit of MHH (118)**

Table 5.1. Overview of ACMG classification tools benchmarked against GenOtoScope.
* Classification not based on ACMG/AMP.
** Classification based on ACMG/AMP guidelines specified for HL, by manual curators.

The unimplemented criteria by `GenOtoScope` are 12. These criteria are: PS2, PS3, PS4, PM3, PM6, PP1, PP4, BS2, BS3, BS4, BP2 and BP5. The main reasons not to implement these criteria are: (i) the lack of established processing algorithm (ii) the lack of data and (ii) further patient information. That is, for the criteria needing functional data, PS3 and BS3, there are no established algorithms that can automatically extract the result of a functional study publication for a given human variant. As the lack of data is concerned, the examination of the PS4 criterion cannot be automated as there is no database to contain the prevalence of affected and control individuals for all possible variant types. Equally, there is no database with the respective information to automate BS2 and BP2 criteria. Last, the need for genomic data from the patient's family disables the examination of the segregation data criteria: PS2, PM3, PM6, PP1, PP4, BS4 and BP5.

The number of missing implemented criteria is competitive with the other classifications tools. VIP-HL implements only one extra criterion, BS2. The main reason not to also implement BS2 is that VIP-HL uses particular thresholds, which are not specified by ACMG HL original work, and thus it may not reflect all penetrance and inheritance modes of all HL-related genes.

InterVar implements 18 out of the 28 ACMG original criteria Richards et al. (2015). As explained in the introduction, disease-specific ACMG criteria may vary from the original 24. Therefore the PP2, PP5, BP1 and BP6 criteria automated by InterVar are not applicable for HL. The remaining two criteria automated by InterVar and not by `GenOtoScope` are PS4 and BS2. To automate these criteria, InterVar used the ANNOVAR annotation tool Wang et al. (2010). However, this tool implements PS4 using a general threshold on a phenotype-based GWAS catalog, consequently the called enriched pathogenic variants may not include all HL-relevant variants. Similarly, to automate BS2 criterion, InterVar uses the zygotic information of a healthy individual in the 1000 Genomes project Clarke et al. (2017) based on the inheritance mode of the variant. Nevertheless, specific thresholds of healthy individuals should be used for HL, which are not published by Oza et al. (2018). As a consequence, there may be false negative cases; InterVar should activate PS4 or BS2 for a given HL variant but it does not. Finally, the remaining criteria need manual curation or additional information not publicly available (e.g. segregation or phenotypic data), therefore they are not implemented by any of the three classification tools.

In our thorough evaluation, shown in the results section, we demonstrate that regarding the 12 ACMG criteria processed by either tool, `GenOtoScope` achieved the best averaged accuracy and precision scores for both tested data sets. This is due to the activation frequency of these criteria being much closer to human curation in `GenOtoScope` than in VIP-HL and InterVar, which trigger the commonly implemented 12 criteria much less frequently. To sum up, our choice to implement these 12 criteria, which are refined for HL, can lead to standardized classification results for all HL-relevant genes.

Besides, our implementation of the criteria presents two more advantages: In contrast to the usage of the ANNOVAR annotation tool, licensed for commercial use, we construct all annotation files needed to examine the ACMG criteria, using freely accessible databases and offer `GenOtoScope` with an open-source software licence. Therefore, any interested researcher can update the corresponding code section to produce adjusted annotations to her needs. Equally, the researcher can update the code to change the steps used to examine a given criterion. The second advantage is that `GenOtoScope` (like VIP-HL) outputs comments for each examined criterion, whereas InterVar does not. This extra information can facilitate the variant curator to justify the activation of a criterion and thus increace the explainability of the classification.

Implementation note

To automate the examination of the ACMG evidence-based criteria we used the Python programming language. First, we used the hgvs library Wang et al. (2018) to parse the variant information for each Ensembl transcript. Then, using the HGVS format we constructed the observed coding sequence and we extracted the needed information for the criteria, through the PyEnsembl library[1]. For example, we extracted the start and stop positions of exons and the positions of the start and stop codons. This library uses the Ensembl version 75 for GRCh37 human genome. We used the PyVCF library[2] to parse variants from VCF files, for instance the ClinVar variants. We applied the Pybedtools Dale et al. (2011) to find the intersections of annotation files, such as the overlap of UniProt domains with repeat regions. Finally, we utilized the BioPython library Cock et al. (2009) for all other tasks for example, to convert cDNA codons to amino acids. `GenOtoScope` currently works only for grch37 genome assembly coordinates. The performance metrics were calculated using scikit-learn library Pedregosa et al. (2011).

The bioinformatics user shall download the whole tool code along with the set of data needed for its execution, e.g. annotation files HL or known variants with high MAF from the github repository. Besides, at this repository, the user can find example configuration files, example input with the corresponding output files and a documentation on how to install and execute `GenOtoScope` on a linux machine or server. Last, to be able to use the variant annotation script, genotoscope_annotate.py, the user needs to install the megSAP application on a docker container, as explained on the respective tool github repository [3].

5.2.2 GenOtoScope Workflow

In the following, the methodology to implement the ACMG evidence-based criteria for congenital hearing loss is explained in five key steps. The conceptual workflow of the web and command line interface (CLI) of `GenOtoScope` is depicted in Fig. 5.3.

In the first step, the user inputs a variant file (.vcf), which, depending on the used interface, may contain a single variant or a larger set of variants of a patient (e.g. full WES data set). Multiple VCFs can be submitted simultaneously.

Next, functional annotation of the VCF takes place using the VeP annotation tool McLaren et al. (2016) through the megSAP bioinformatics application [4]. The resulting intermediate variant file is organized as a standard matrix file (tabular file) where each row is a variant and a column contains variant annotation. These columns contain the basic variant information (for example chromosome position of

[1] https://github.com/openvax/pyensembl
[2] https://github.com/jamescasbon/PyVCF
[3] https://github.com/imgag/megSAP
[4] https://github.com/imgag/megSAP

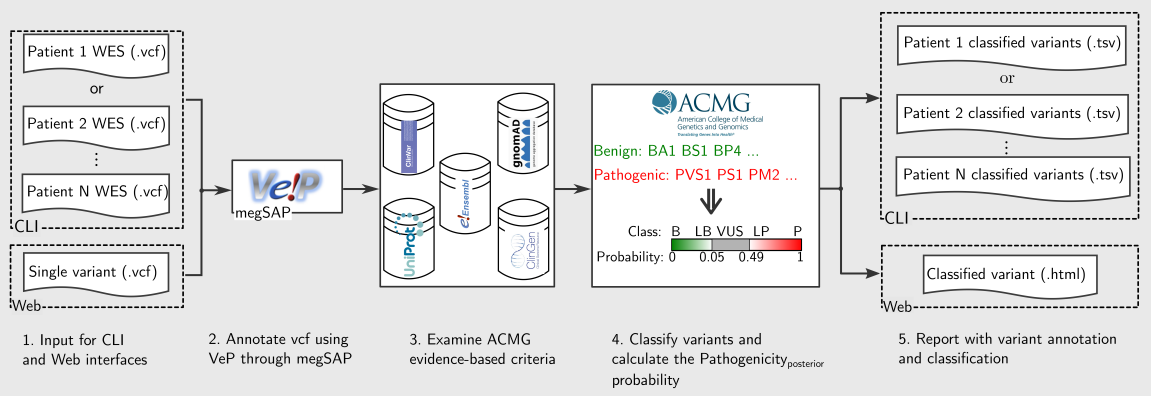


Figure 5.3. Conceptual workflow of GenOtoScope.

variant and affected gene name), the transcript and the protein HGVS signature and the unique functional annotation for the variant (e.g. minor allele frequency in gnomAD subpopulations Karczewski et al. (2020), the OMIM variant observed clinical description Hamosh et al. (2000) and the REVEL pathogenicity score for the variant Ioannidis et al. (2016)).

The third step uses the core sub-algorithms of `GenOtoScope` to automatically analyze the listed variants according to ACMG criteria. These sub-algorithms access programmatically four databases: the human clinical variants database ClinVar Landrum et al. (2018), the human exomes database gnomAD Karczewski et al. (2020), the protein knowledge database UniProt Consortium (2021) and the clinical genome database Rehm et al. (2015). Extracted annotations are organized based on the Ensembl features Howe et al. (2021) for a variant-affected transcript. Beyond the mere result of checking a criterion (activation or non-activation), the tool stores a descriptive comment on the reason for activation or non-activation, to be used as an explanation for the user.

In the following step, the tool combines the activated evidence-based criteria to classify the variant into 5 pathogenicity categories ("benign", "likely benign", "VUS", "probably pathogenic" and "pathogenic") according to ACMG guidelines. If none of the criteria is activated, the tool classifies the variant as VUS. Subsequently, in the same fourth step, `GenOtoScope` computes the pathogenicity posterior probability based on Tavtigian et al. (2018). This is intended to allow a better discrimination of VUS and additional re-classification of VUS into benign or pathogenic variants.

In the fifth and final step, `GenOtoScope` extends the intermediate annotation tabular file with the criteria activation results and the comments along with the predicted ACMG class and the computed pathogenicity probability. Finally, the tool will save this file as the produced classification output.

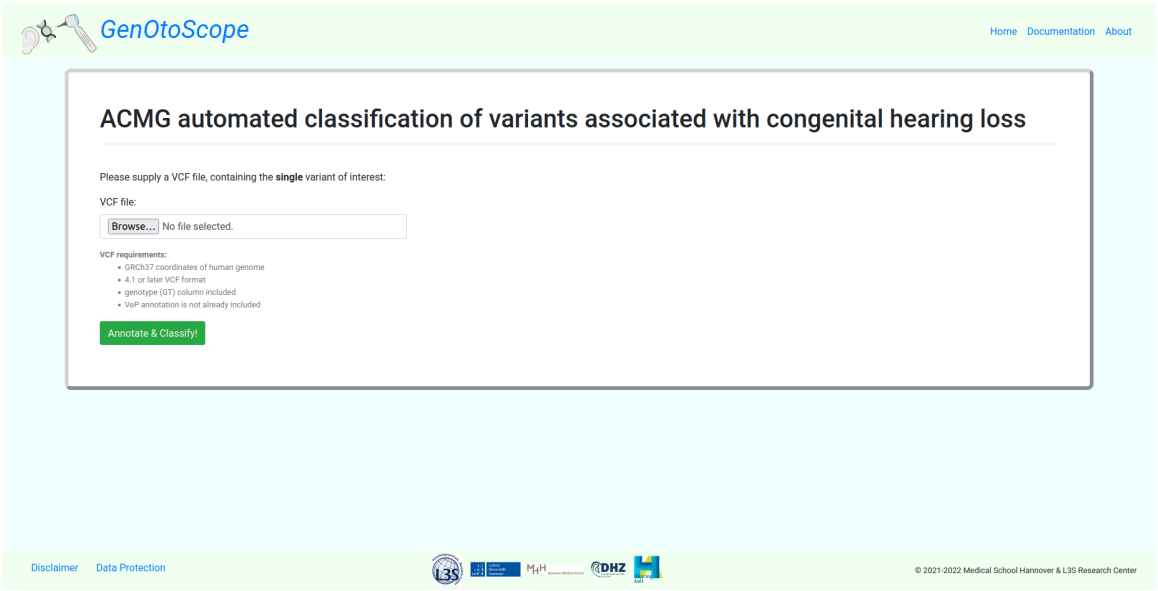
A crucial sub-step of this workflow is the construction of annotation files, which is needed for the automatization of the examination of the ACMG evidence-based

criteria. We constructed the needed annotation files, clinical-significant exons, HL-relevant transcripts, critical regions for proteins, critical regions for proteins without benign variants and protein repeat regions without domain intersection, using publicly available data sets.

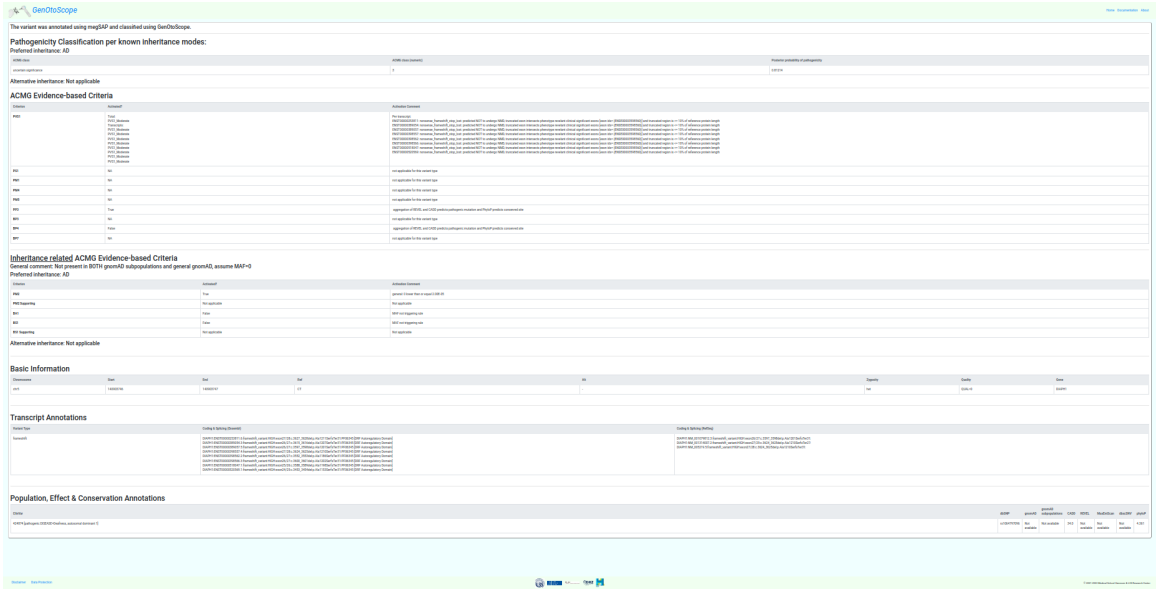
5.2.3 GenOtoScope Interfaces

Web Interface

The web application is targeted for free online usage. Advanced bioinformatics skills are not required. A screenshot of the home page of the `GenOtoScope` website is shown in Fig. 5.4. Users can upload a single variant file (.vcf). The website will annotate and convert the VCF to GSvar file through the megSAP application. A result page (.html) will be generated to show the basic annotation of the variant, its ACMG classification and the computed pathogenicity posterior probability.



(a)



(b)

Figure 5.4. Web interface of GenOtoScope. (a): The home page of the `GenOtoScope` website. (b): The output page, for an example variant (RS id: 1064797096), which includes its classification based on HL-specified ACMG guidelines.

Command Line Interface

The command line interface (CLI) is tailored to bioinformatics personnel. The first command of this mode, `genotoscope_annotate.py`, will accept as input a folder

of VCF files or a single VCF file. It will annotate the input VCF files and convert them into GSvar files. The second command, `genotoscope_classify.py`, will accept as input a folder of GSvar files or a single GSvar file, the output of the previous command. Then it will automatically examine the ACMG evidence-based criteria to classify and compute the pathogenicity posterior probability for each variant in an input GSvar file. The output will be an extended GSvar file containing information on the examination of the ACMG evidence-based criteria, the ACMG pathogenicity class and the pathogenicity posterior probability. Examples of these two commands are shown in Fig. 5.5.

```
python3.7 genotoscope_annotate.py
--input_variants /home/username/example_vcfs
--output_path /home/username/example_vcfs_annotate
--docker_root /megSAP
--docker_root_host /home/username/dockerspace
--docker_image_id XYZ
--logging INFO
```

(a)

```
python3.7 genotoscope_classify.py
--analysis_root /home/username/analysis
--input_variants /home/username/example_gsvars
--output_path /home/username/example_gsvars_classify
--settings_file /home/username/analysis/genotoscope_server.yaml
--filter_HL_genes 0
--min_pathogenicity 0.49
--logging INFO
--threads 1
```

(b)

Figure 5.5. Command line examples for the two commands of GenOtoScope. (a) Annotate all variants presented in VCF files, in input folder, using megSAP application and save results in GSvar files. (b) Classify all variants presented in GSvar files based on ACMG guidelines specified for HL.

5.2.4 Automating Examination of ACMG Evidence-based Criteria

In the following subsections, we briefly describe our implementation of the aforementioned 12 ACMG criteria: PVS1 is automated based on Abou Tayoun et al. (2018). Information from ClinVar database is used for the implementation of PS1 and PM5 (including PM5 Strong). Automation of PM1 examines critical regions provided by Oza et al. (2018) and a purpose-built annotation file containing critical regions without benign mutations. Customized annotation files are also used for (non) repetitive region dependent criteria PM4 and BP3, whereas automation of PP3, BP4 and BP7 employs established prediction algorithms. Population frequency data for implementation of PM2 (PM2 Supporting), BA1 and BS1 (BS1 Supporting) is taken from gnomAD database.

Refined PVS1

PVS1 criterion is assessed for start-loss, nonsense (stop gained), stop-loss, frameshift, in-frame, splice acceptor and donor variants according to Abou Tayoun et al. (2018).

First, the occurrence of nonsense-mediated decay (NMD) is predicted by a subroutine for each affected transcript using the HGVS signature of the variant to create the observed coding sequence per exon. Altered region is defined as variant-affected

coding region. The algorithm locates the 5'-closest stop codon and follows the scheme of Hu et al. (2017) to assess impact of this premature termination codon (PTC) on NMD: Observed coding sequence is rated to escape NMD if PTC appears either within the 50 last bases of the penultimate exon or at most 200 bases downstream from the start codon or alternatively the transcript contains no introns. Otherwise, NMD is classified to occur. Fig. 5.6 illustrates this subprocess.

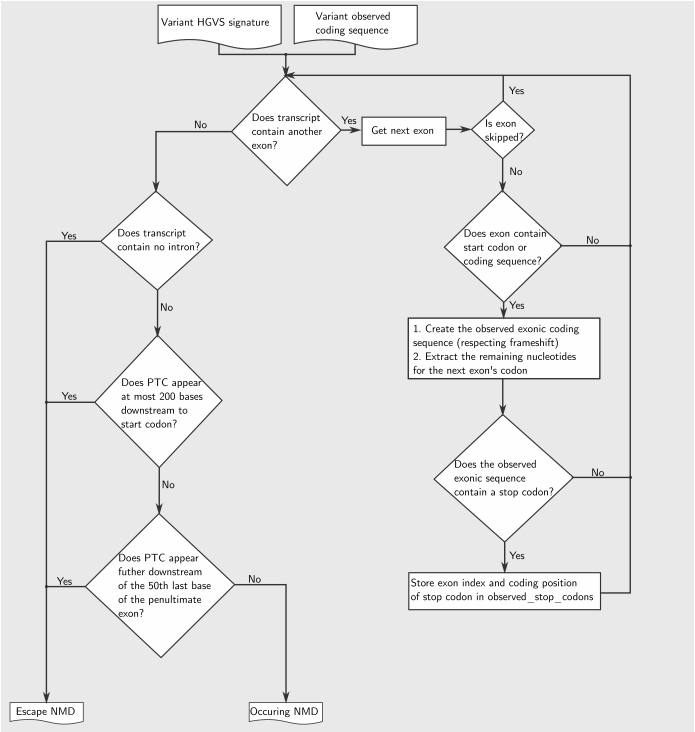


Figure 5.6. Conceptual flowchart to assess NMD for the refined PVS1 rule.

If NMD is predicted to occur, the algorithm intersects the stored variant-affected coding region to phenotype-relevant transcripts to decide the PVS1 outcome. If NMD is not predicted to occur, it intersects the variant-affected coding region with protein domain regions. If there is an intersection, it examines if the affected region overlaps a critical domain for protein function, to decide the PVS1 outcome. If the affected region is not within a known domain, overlap with clinically significant exons and phenotype-relevant transcripts is examined. If this is confirmed, it is investigated whether the PTC results in the removal of more than 10% of the reference protein product.

For start-loss variants, the algorithm first checks if any other transcript contains an alternative start codon. If not, it extracts potential in-frame start codons that are no further than 200 bases downstream of the lost start codon. Next, it queries ClinVar for pathogenic entries with at least one review star between the lost start codon and the detected in-frame start codon. If there is such a ClinVar entry, PVS1

(Moderate) is triggered, otherwise PVS1 (Supporting).

PS1 and PM5 (PM5 Strong)

The workflow of assessing the PS1 and PM5 criteria is shown in Fig. 5.7.

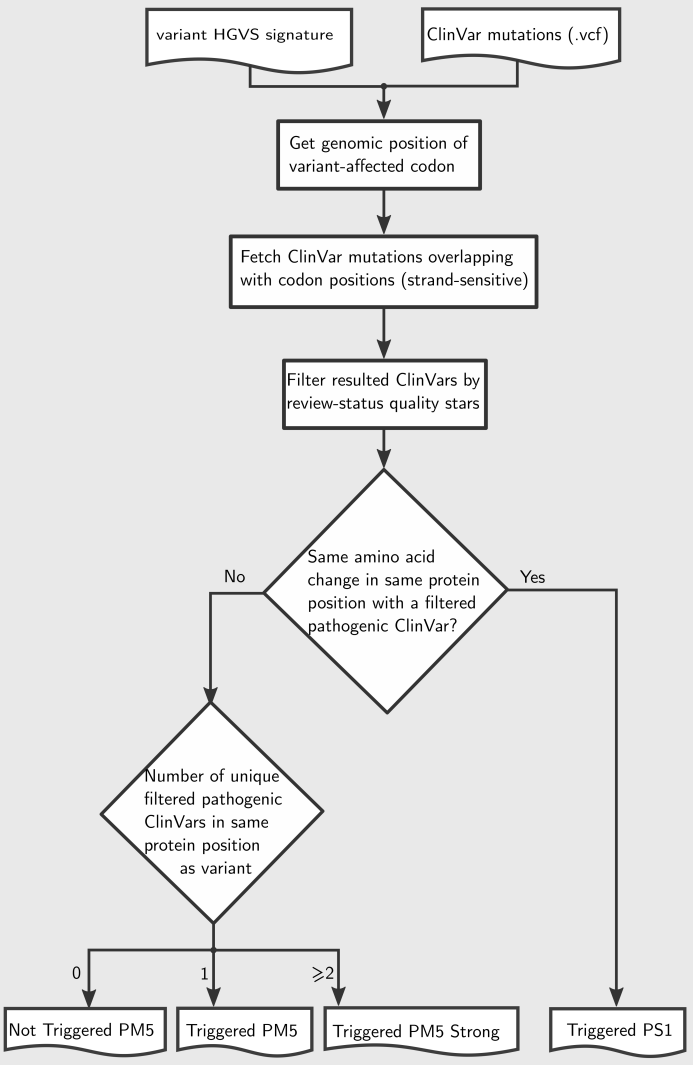


Figure 5.7. Conceptual flowchart for examining PS1 and PM5 (PM5 Strong).

First, genomic positions of the affected codon are computed based on exonic variant location and directionality of the respective gene. Then, all missense variants at corresponding genomic positions are extracted from ClinVar and filtered by strand to match the directionality of the affected gene. Additionally, ClinVar entries can be filtered by review status as the user can define a minimum number of quality stars as a threshold for variants to be considered (default value: 1).

The filtered variants and resulting amino acids are used for further assessment of PS1 and PM5 criteria: PS1 is triggered if any filtered-in variant from ClinVar that is rated as pathogenic results in the same amino acid change as the observed variant. PM5 is triggered if the filtered-in ClinVar entries do not contain the observed amino acid change but at least one pathogenic variant affecting the same codon. If the entries include two or more such variants, PM5 is applied as strong evidence (PM5 Strong) according to Oza et al. (2018).

PM1

For automation of PM1 a custom-made annotation file is used. It comprises all critical protein regions without benign ClinVar entries and also includes specific domains and motifs of hearing loss related proteins as defined by Oza et al. (2018). Precisely, these regions are the pore-forming domain of KCNQ4 gene and the three-stranded helices of the collagen genes *COL11A2*, *COL4A3*, *COL4A4* and *COL4A5*. PM1 is applied to missense variants overlapping any of the annotated genomic regions. If the variant overlaps on the three-stranded motifs of the collagen genes, it accepts only the matches that affect the Glycine residues contained in a Gly-X-Y motif.

The precise regions, used for PM1 criterion, are the pore-forming domain of KCNQ4 gene and the three-stranded helices of the collagen genes COL11A2, COL4A3, COL4A4 and COL4A5. PM1 is applied to missense variants overlapping any of the annotated genomic regions. If the variant overlaps on the three-stranded motifs of the collagen genes, it accepts only the matches that affect the Glycine residues contained in a Gly-X-Y motif.

PM4

The algorithm for this criterion is applied to all in-frame (deletions/duplications) and stop-loss variants that do not trigger PVS1 in any strength level. Considering PTC assessed in PVS1 subroutine, length of observed proteins is calculated and compared to reference protein length. For length differences >10%, PM4 is triggered except for variants in known repetitive regions derived from UniProt.

BP3

The algorithm for this criterion is applied to all in-frame (deletions/duplications) and nonsense (stop gained) variants. Based on an annotation file containing functional domains and repeat regions derived from UniProt, BM3 is triggered if the variant-affected coding region overlaps a repetitive region without known function.

PP3, BP4 and BP7

GenOtoScope incorporates in silico tools for conservation (PhyloP Cooper et al. (2005)), splicing (MaxEntScan Yeo and Burge (2004), dbscSNV Jian et al. (2014)) and missense-prediction (REVEL Ioannidis et al. (2016), CADD Rentzsch et al. (2019)).

For missense variants, activation of PP3 requires positive pathogenicity prediction (REVEL/CADD) and high conservation (PhyloP) scores. In contrast, BP4 is triggered if conservation and predicted probability of pathogenicity are low and moreover, the variant is also estimated not to affect splicing (MaxEntScan/dbscSNV).

Variants with no immediate impact on amino acid sequence (exclusion: canonical splice site variants) are similarly screened for potential effects on splicing. If splicing is predicted to be affected and the nucleotide is highly conserved, PP3 is activated. Conversely, if a potential splice variant is predicted to have no splicing effect and conservation is low, BP7 is triggered for synonymous variants and BP4 for other variant types respectively.

The used thresholds by prediction follow. To decide upon pathogenicity, we aggregated CADD and REVEL in the following scheme: if CADD score is greater than 20, then we set $\text{CADD}_{\text{vote}} = 1$ otherwise $\text{CADD}_{\text{vote}} = 0$. For REVEL, if REVEL score is greater or equal to 0.7, then $\text{REVEL}_{\text{vote}} = 1$, else if REVEL score is lower or equal to 0.15, then $\text{REVEL}_{\text{vote}} = 0$, otherwise we set $\text{REVEL}_{\text{vote}} = 0.5$. Finally, if the average voting of $\text{CADD}_{\text{vote}}$ and $\text{REVEL}_{\text{vote}}$ is greater or equal to 1, GenOtoScope assumes that the variant is pathogenic. For splicing impact, we aggregate the predictors MaxEntScan and dbscSNV in the following scheme: if $|\frac{\text{observed}_{\text{score}} - \text{reference}_{\text{score}}}{\text{reference}_{\text{score}}}|$ is greater than 0.15, then $\text{MaxEntScan}_{\text{vote}} = 1$ otherwise $\text{MaxEntScan}_{\text{vote}} = 0$. For dbscSNV, if either ADA score or RF score is greater than 0.6 then $\text{dbscSNV}_{\text{vote}} = 1$, otherwise $\text{dbscSNV}_{\text{vote}} = 0$. We aggregate the votes similarly to pathogenicity. That is, if the average voting of $\text{MaxEntScan}_{\text{vote}}$ and $\text{dbscSNV}_{\text{vote}}$ is greater or equal to 1, GenOtoScope decides that the variant has a splicing impact. Last for conservation prediction, we used PhyloP score, as follows: if PhyloP score is greater of 1.6 then GenOtoScope decides that this is a conserved site, otherwise GenOtoScope decides that the site is not a conserved site.

PM2 (PM2 Supporting) BA1 and BS1 (BS1 Supporting)

Assessment of population data criteria uses adjustable minor allele frequency (MAF) thresholds, which by default are the ones defined by Oza et al. (2018). Each gene can be assigned a preferred mode of inheritance, which can be customized by providing an input file. Default settings comprise the inheritance modes of 164 hearing loss gene-disease pairs defined by the ClinGen Hearing Loss Gene Curation Expert Panel DiStefano et al. (2019) plus preferred inheritance patterns for additional genes specified by the HG department of MHH. We will refer to ClinGen Hearing Loss Gene

Curation Expert Panel commitee as VCEP-HL for convenience.

For each variant, allele frequencies (AF) of gnomAD subpopulations are retrieved. Known pathogenic variants with high AF are excluded from further assessment of BA1 and BS1 according to Oza et al. (2018). AF of each subpopulation and the median AF of all subpopulations are evaluated with respect to the appropriate inheritance mode threshold. PM2 (PM2 Supporting), BA1 and BS1 (BS1 Supporting) are triggered, if any subpopulation's AF or the median AF matches the respective inheritance mode threshold.

Regarding different inheritance patterns, the algorithm by default utilizes distinct thresholds for autosomal dominant and autosomal recessive inheritance mode as specified by Oza et al. (2018). For the X-linked mode of inheritance, autosomal dominant thresholds are adopted. If no mode of inheritance is provided, it is assumed to be unknown. In these cases, the algorithm selects the strictest threshold between autosomal dominant and recessive for each criterion. For mitochondrial genes, the same procedure is used as for unknown mode of inheritance, with an additional warning, since the application of ACMG criteria is validated only for Mendelian disorders.

Hearing-loss Specific ACMG Classification

Having assessed all applicable criteria for a given genomic variant, GenOtoScope combines the activated criteria to compute the respective ACMG class using the five-tier terminology system ("benign", "likely benign", "VUS", "likely pathogenic" and "pathogenic") defined by Richards et al. (2015).

Moreover, `GenOtoScope` incorporates the extended recommendations of VCEP-HL for the following criteria combinations: (i) Variants triggering PVS1 and PM2 (Supporting) will be classified as "likely pathogenic" for genes associated with autosomal recessive inheritance. (ii) Variants activating BS1 without triggering any pathogenic criterion will be classified as "likely benign".

Computation of Pathogenicity Probability

After having classified all exome variants of a patient, a number of variants are classified as VUS, due to insufficient or conflicting triggered evidence criteria. To help the human curators to discriminate the pathogenicity of the VUS cases in a quantitative manner, `GenOtoScope` calculates the pathogenicity probability for each variant following Tavtigian et al. (2018). The calculation of the pathogenicity probability is calculated automatically for all input variants.

`GenOtoScope`applies the naive Bayes model to calculate the posterior probability of

pathogenicity given the triggered ACMG evidence rules using the following equations:

$$\text{Pathogenicity}_{\text{posterior}} = \frac{\text{Pathogenicity}_{\text{likelihood}} \cdot \text{Pathogenicity}_{\text{prior}}}{(\text{Pathogenicity}_{\text{likelihood}} - 1) \cdot \text{Pathogenicity}_{\text{prior}} + 1} \tag{5.1}$$

$$\text{Pathogenicity}_{\text{likelihood}} = \text{O}_{\text{PVST}}{}^{(\frac{\text{N}_{\text{PSU}}}{8} + \frac{\text{N}_{\text{PM}}}{4} + \frac{\text{N}_{\text{PST}}}{2} + \frac{\text{N}_{\text{PVST}}}{1} - \frac{\text{N}_{\text{BSU}}}{8} - \frac{\text{N}_{\text{BST}}}{2})}, \tag{5.2}$$

where the default parameters are used: $\text{Pathogenicity}_{\text{prior}} = 0.1$, $\text{O}_{\text{PVST}} = 350$ and $\text{X} = 2$.

The calculation of the pathogenicity probability is calculated automatically for all input variants.

5.3 Experimental Evaluation

5.3.1 Variant Classification

Data Sets

`GenOtoScope` variant classification was compared to similar tools: (1) InterVar, a tool for variant classification tested across a spectrum of phenotypes Li and Wang (2017); (2) VIP-HL, the recently published tool for hearing loss Peng et al. (2020). We benchmarked the accuracy and precision of variant classification on two data sets.

The first data set is the publicly available set of manually annotated variants by ClinGen VCEP-HL Patel et al. (2021), hereafter referred to as VCEP-HL data set. This data set contains manual annotation for 158 variants associated with HL. These variants are contained in 9 HL-relevant genes (*USH2A*, *COCH*, *GJB2*, *KCNQ4*, *MYO7A*, *MYO6*, *TECTA*,*SLC26A4* and *CDH23*). The second data set is the private set of manually annotated variants by the HG department of MHH, hereafter referred to as MHH data set. The MHH data set contains 118 variants, contained in 36 HL-relevant genes. More specifically, the included genes are: *COL11A1*, *USH2A*, *NLRP3*, *OTOF*, *ALMS1*,*PAX3*, *ILDR1*, *WFS1*, *COL11A2*, *COL9A1*, *MYO6*, *SLC26A4*, *CHD7*, *GRHL2*, *TMC1*, *WHRN*, *TNC*, *MYO3A*, *PCDH15*, *CDH23*, *OTOG*, *MYO7A*, *TECTA*, *COL2A1*, *MYO1A*, *P2RX2*, *GJB2*, *GJB6*, *ACTG1*, *MYH14*, *KCNE1*, *TMPRSS3*, *MYH9*, *SOX10*, *POU3F4* and *PRPS1*.

Performance Metrics

To assess the prediction performance, we combine "benign" and "likely benign" classes to "Benign", "pathogenic" and "likely pathogenic" classes to "Pathogenic". Thus, we created a three-class prediction task, containing the "Benign", "Pathogenic" and "VUS" as the three possible broader classes.

Following the evaluation of the classification tool TAPES citexavier2019tapes, we evaluated the accuracy and precision of each software tool, calculating the area under

the curve (AUC) of the Receiver Operating Characteristics (ROC) curve and of the precision-recall curve, accordingly.

Refined Classification of VUS

We acknowledge that not all evidence-based criteria for HL can be automated, due to the need of further patient's genomic information and the need for manual curation of certain criteria. Therefore, `GenOtoScope` currently implements 12 out of the 24 ACMG criteria for HL. Thus, `GenOtoScope`, which uses the standard classification scheme (Fig. 5.2), will misclassify a variant as VUS even if it belongs in the broader classes of "Benign" or "Pathogenic".

To investigate the `GenOtoScope` classification potential, we provided a refined classification of variants, classified as "VUS", by original `GenOtoScope`, based on calculated pathogenicity probability and not the mere classification scheme, following the idea from TAPES classification tool Xavier et al. (2019). We will refer to this refined version of `GenOtoScope`, as `GenOtoScope_prob`.

Based on Tavtigian et al. (2018), the range of values of the pathogenicity probability would be lowered, if a subset of the original ACMG criteria were automatized. Thus, the range of values of the pathogenicity probability calculated for a variant, using the automated criteria by `GenOtoScope`, will be reduced, compared to the classification provided, by a manual curator, who has evaluated all possible ACMG criteria.

To this end, `GenOtoScope` pathogenicity probability reclassified the VUS variants, classified by the `GenOtoScope`, by their calculated pathogenicity probability in the following Alg. 4 :

Algorithm 4 `GenOtoScope_prob`

```
function REFINE_VARIANTS_OF_UNCERTAIN_SIGNIFICANCE(predicted_class_genotoscope, pathogenicity_posterior_genotoscope)
    if predicted_class_genotoscope = "VUS" then
        if predicted_posterior_genotoscope ≥ 0.49988 then
            refined_class ← "Pathogenic"
        else if predicted_posterior_genotoscope ≤ 0.05072 then
            refined_class ← "Benign"
        else
            refined_class ← "VUS"
        end if
    else                                   ▷ Do not refine "Benign" or "Pathogenic" classifications
        refined_class ← predicted_class_genotoscope
    end if
    return refined_class
end function
```

For an immediate comparison with the probability threshold if all criteria were implemented, see Fig. 5.2.

We have chosen these threshold values, based on relaxing the lowest combination of the triggered criteria needed to predict either one of the broader classes of "Pathogenic" or "Benign" based on Richards et al. (2015) and Oza et al. (2018). Con-

sequently, the selection of these thresholds is not dependent on a given test data set, but on the currently implemented ACMG criteria for HL.

Then we have transformed this relaxed combination of criteria to pathogenicity probability based on Eq. 5.1.

That is, for the "Pathogenic" broader class, the combination of the criteria, with the least pathogenicity strength, resulting in "likely pathogenic" class, is "1 pathogenic moderate criterion and at least 4 pathogenic supporting criteria" (Richards et al. (2015) and Oza et al. (2018)). However, based on available open data and further patient genetic data we have implemented seven out of the fourteen ACMG criteria favouring the "Pathogenic" broader class. Therefore we lowered the combination to "1 Moderate and 1 Supporting criterion" which translated to the probability of 0.49988. Therefore, the `GenOtoScope_prob` will refine the "VUS" class, by the original `GenOtoScope`, to "Pathogenic" for a variant with pathogenicity probability equal to at least 0.49988.

Similarly, for the "Benign" broader class, the combination of criteria with the lowest strength is "at least two benign supporting criteria" and results in the "likely benign" class. For the same reasons, the `GenOtoScope` currently implements five out of the total ten applicable criteria for the "Benign" broader class. Therefore we reduce the requirements of this combination to be "one benign supporting criteria" which translates to the pathogenicity probability of 0.05072. Consequently, `GenOtoScope` will reclassify a variant classified as "VUS", by `GenOtoScope`, to "Benign" broader class if the variant's probability is lower or equal to 0.05072.

Investigation of Performance Discrepancies

We sought out to investigate the reasons for the discrepancy in prediction performance between the classification tools. To do so, we extended the troubleshooting plots of Nicora et al. (2018), by calculating the log ratio of the activation frequency of an evidence-criterion by a classification tool and the manual curation, as:

$$\mathrm{r}_{\mathrm{k}}^{\mathrm{e,c}} = log_{10}(\frac{\alpha_k^{e,c}}{\alpha_{\mathrm{manual}}^{e,c}}), \tag{5.3}$$

where $\alpha_k^{e,c}$ is the activation frequency of e, any of the implemented ACMG rules, by a tool $k = \{\text{InterVar, VIP-HL, GenOtoScope}\}$ for a broader class $c = \{\text{pathogenic, VUS, benign}\}$.

We computed all log ratios for each evidence rule, e, by each classification tool for the three grouped classes, c. Finally, we used heatmap plots to depict these log ratios.

5.3.2 VCEP-HL Data Set

The ROC and precision-recall curves are shown in Fig. 5.8 and Fig. 5.9, respectively. We observe that GenOtoScope and GenOtoScope pathogenicity probability achieved

the best AUC scores for all three classes. In Precision-recall curves, VIP-HL achieved slightly higher AUC compared to GenOtoScope for the benign class. However, for the other two classes again GenOtoScope and GenOtoScope pathogenicity probability achieved the best AUC scores. Besides, we calculated the performance scores, AUC of ROC and the average precision of the precision-recall curves for all classification tools. We show the micro-averaged scores, over the three broader classes ("Benign", "VUS", "Pathogenic"), are shown in Table 1. Based on this table, the two versions of GenOtoScope classification achieved the best results for both AUC of ROC and the average precision.

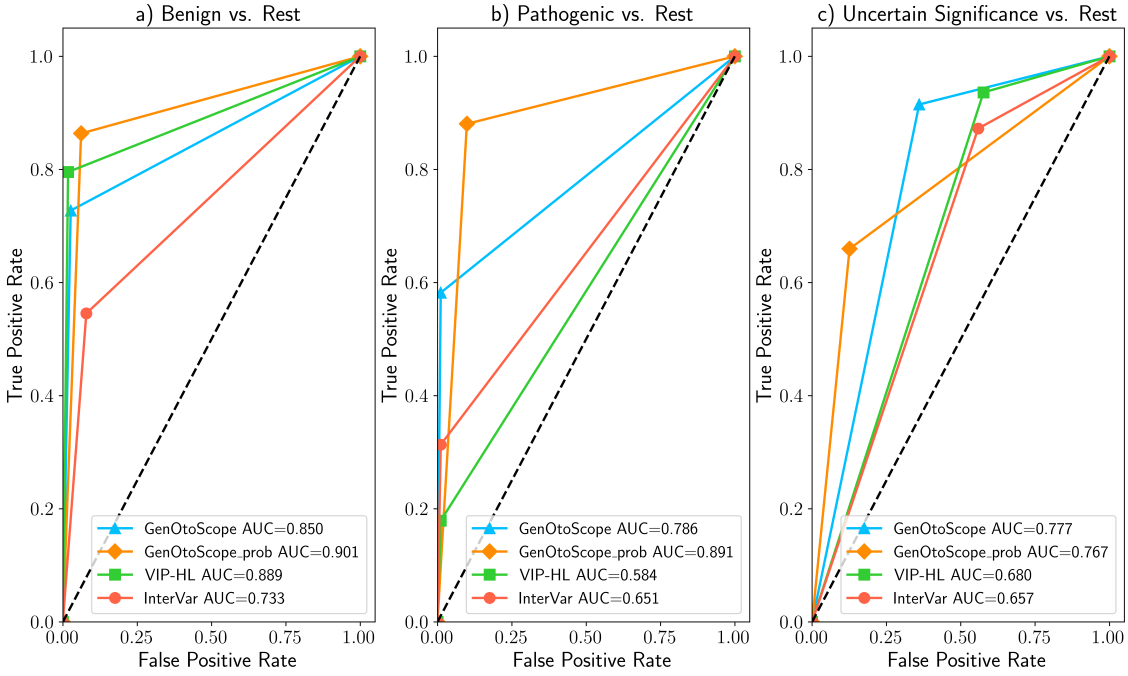


Figure 5.8. ROC curves and AUC scores of all classification tools for VCEP-HL data set: (a) Prediction of the "Benign" broader class versus the "Pathogenic" broader class and the VUS class (b) Prediction of the "Pathogenic" broader class versus "Benign" broader class and the VUS class (c) Prediction the "VUS" class versus the "Benign" broader class and the "Pathogenic" broader class.

Additionally, we calculated the performance scores, AUC of ROC and the average precision of the precision-recall curves for all classification tools. We show the micro-averaged scores, over the three broader classes ("Benign", "VUS", "Pathogenic") in Table 5.2. Based on this table, the two versions of `GenOtoScope` classification achieved the best results for both AUC of ROC and the average precision.

To explain the difference in prediction performance, we plot the heatmaps of the log ratio of activation frequency between a classification tool and the manual curation (5.3). The results are shown in Fig. 5.10.

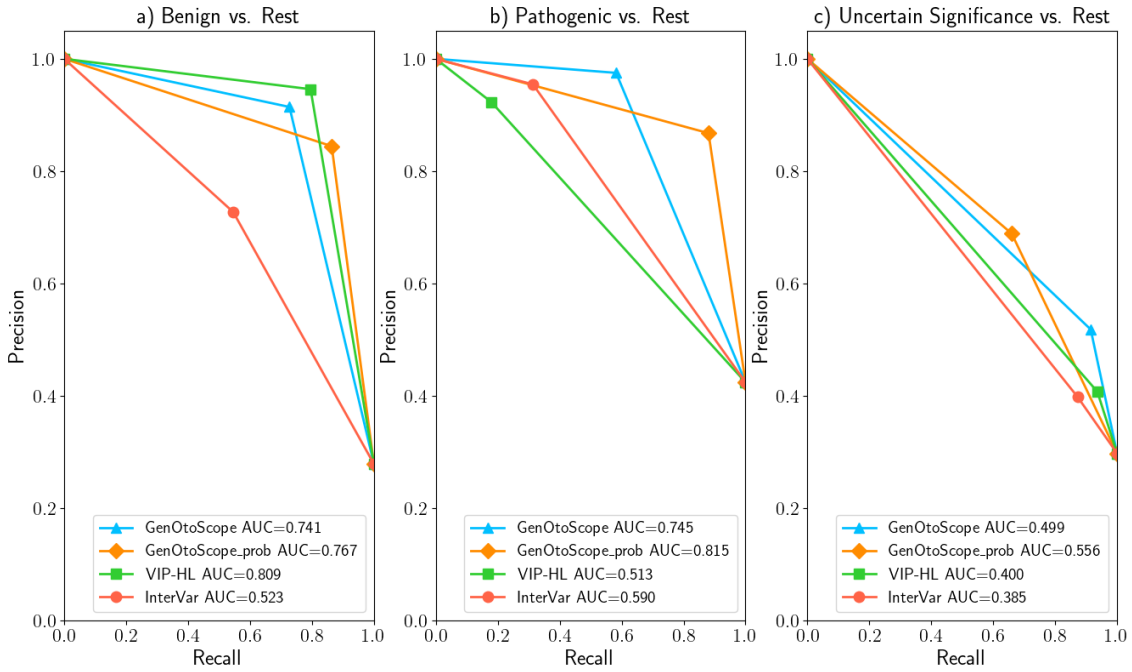


Figure 5.9. Precision-recall curves and AUC scores of all classification tools for VCEP-HL data set: (a) Prediction of "Benign" broader class versus the "Pathogenic" broader class and the VUS class (b) Prediction of the "Pathogenic" broader class versus "Benign" broader class and the VUS class (c) Prediction of the "VUS" class versus the "Benign" broader class and the "Pathogenic" broader class.

Performance scores	**Classification Tools**			
	`GenOtoScope`	`GenOtoScope_prob`	VIP-HL	InterVar
ROC AUC	0.79114	**0.85759**	0.68196	0.65823
Average Precision	0.61342	**0.71960**	0.47307	0.44817

Table 5.2. Micro-averaged performance scores for all classification tools, over the three broader classes in the VCEP-HL data set. Best values of a performance score, across all classification tools, are shown in bold.

We observed the following patterns for each grouped class. First, for the "Pathogenic" broader class, VIP-HL activated 8 implemented pathogenic rules (PVS1 (Strong), PVS1 (Moderate), PM1, PM5, PVS1 and PM2) from 32 times less (PVS1 (Strong)) to 79 times less (PM2) than the manual curation. Nevertheless, `GenOtoScope` activated 5 out of these 8 rules with the same frequency as the manual curation (PVS1, PM2, PP3, PM2 (Supporting) and PM5). It activated the remaining 3 rules (PVS1 (Moderate), PM1 and PVS1 (Strong)) approximately twice as much as the manual curation.

For the "VUS" class, we observed that VIP-HL activated 8 implemented rules

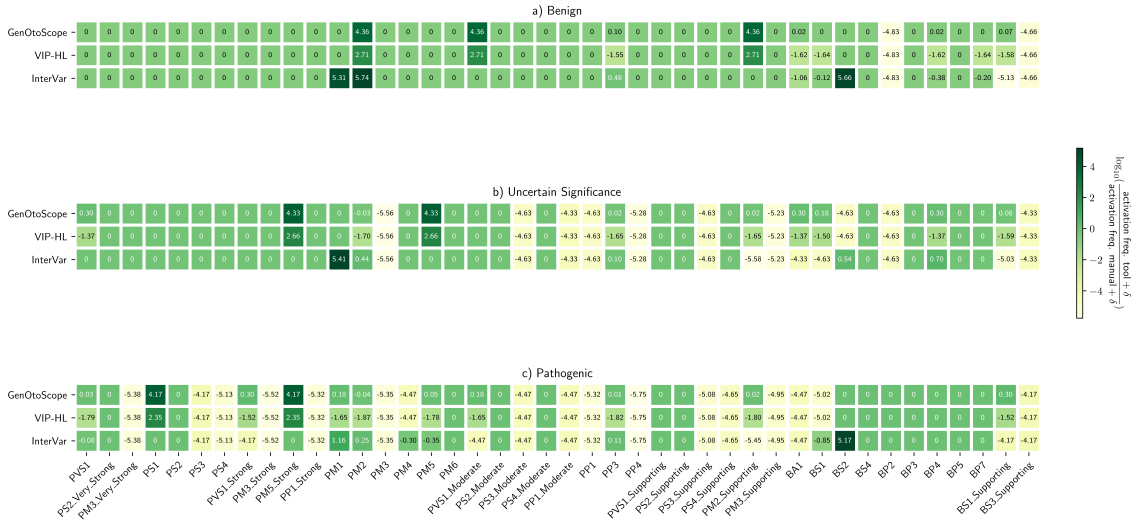


Figure 5.10. Activation frequency ratios for VCEP-HL data set. Log ratios calculated for each of the three classes classified by the VCEP-HL: (a) the "Benign" broader class, (b) the "VUS" class and (c) the "Pathogenic" broader class.

(BP4, BA1, PVS1, BS1, PM2 (Supporting), PP3 and PM2) from 25 times less (BP4) to 50 times less (PM2) than the manual curation. In contrast, `GenOtoScope` activated 3 out of the 8 rules (PM2, PP3, PM2 (Supporting)) with the same frequency as the manual curation and it activated the remaining 5 rules (BS1 (Supporting), BS1, BA1, BP4, PVS1) approximately one to two times more frequently than the manual curation.

For the "Benign" broader class, VIP-HL activated 6 implemented rules (PP3, BS1 (Supporting), BP7, BP4, BS1 and BA1) from 32 times less (BA1, BS1, BP4, BP7, BS1 (Supporting)) to 40 times less (PP3) than the manual curation. `GenOtoScope` activated 4 of these rules (BS1, BP7, BA1, BP4) with approximately the same frequency as the manual curation. The other two rules (PP3 and BS1 (Supporting)) were activated by genotoscope , one time more frequently than the manual annotation.

To examine the reasons for the lower precision of GenOtoScope for the "Benign" broader class compared to VIP-HL, despite GenOtoScope comparable activation frequency of criteria with manual curation, we examined GenOtoScope misclassifications in this class. That is, we examined the variants belonging in the VUS class and misclassified in the "Benign" broader class by GenOtoScope. These misclassified variants are seven, a significant amount for the calculation of precision score, due to the total of 44 variants in the "Benign" broader class. The main reason for the misclassification was that manual annotation used criteria not implemented by GenOtoScope to classify these variants as "VUS". More specifically, the manual curation used criteria which need manual investigation or not available patient's family genomic data (for example PP1, PP4 or PM3), to classify the five out of these seven variants as "VUS".

The last two variants were misclassified by GenOtoScope pathogenicity probability as their calculated probability was lower than the set threshold for refining a VUS as a variant in the "Benign" broader class. VIP-HL could classify correctly four out of these seven variants.

To examine the reasons for the lower precision of `GenOtoScope` for the "Benign" broader class compared to VIP-HL. We examined `GenOtoScope` misclassifications for this grouped class. That is, we examined the variants belonging in the VUS class and misclassified in the "Benign" broader class by `GenOtoScope`. These misclassified variants are seven, a significant amount on the calculation of precision for the total of 44 variants in the "Benign" broader class. The main reason for the misclassification was that manual annotation used unimplemented rules to classify these variants as VUS. More specifically, for five out of the seven variants the manual curation used rules not implemented by `GenOtoScope` (for example PP1, PP4 or PM3) to classify the variants as VUS. The last two variants were misclassified by `GenOtoScope pathogenicity probability` as their calculated probability was lower than the set threshold for refining a variant classified from "VUS" into the "Benign" broader class. For completeness, VIP-HL classified correctly, as "VUS", four out of these seven variants.

5.3.3 MHH Data Set

The ROC curve and AUC scores are shown in Fig. 5.11 and Fig. 5.12, respectively. In ROC curves, `GenOtoScope` or `GenOtoScope pathogenicity probability` scored the highest performance values, compared to VIP-HL and InterVar, for all three classes. In the Precision-Recall curves, `GenOtoScope` outperformed all other classification tools, in terms of AUC score, for benign classification. `GenOtoScope` and `GenOtoScope pathogenicity probability` outperformed all classification tools, in AUC score for pathogenic and VUS classes.

We calculated the micro-average AUC of ROC curves and average precision of Precision-Recall curves, across the three broader classes for each classification tool. We show the results in Table 5.3. As in the previous data set, the two versions of the `GenOtoScope` classification achieved the best scores for both performance metrics.

Performance scores	**Classification Tools**			
	`GenOtoScope`	`GenOtoScope_prob`	VIP-HL	InterVar
ROC AUC	0.88701	**0.90395**	0.86441	0.77966
Average Precision	0.73212	**0.76864**	0.68544	0.53085

Table 5.3. Micro-averaged performance scores for all classification tools, over the three broader classes in the MHH data set. Best values of a performance score, across all classification tools, are shown in bold.

To explain the discrepancy in performance scores, we plotted the heatmap of log

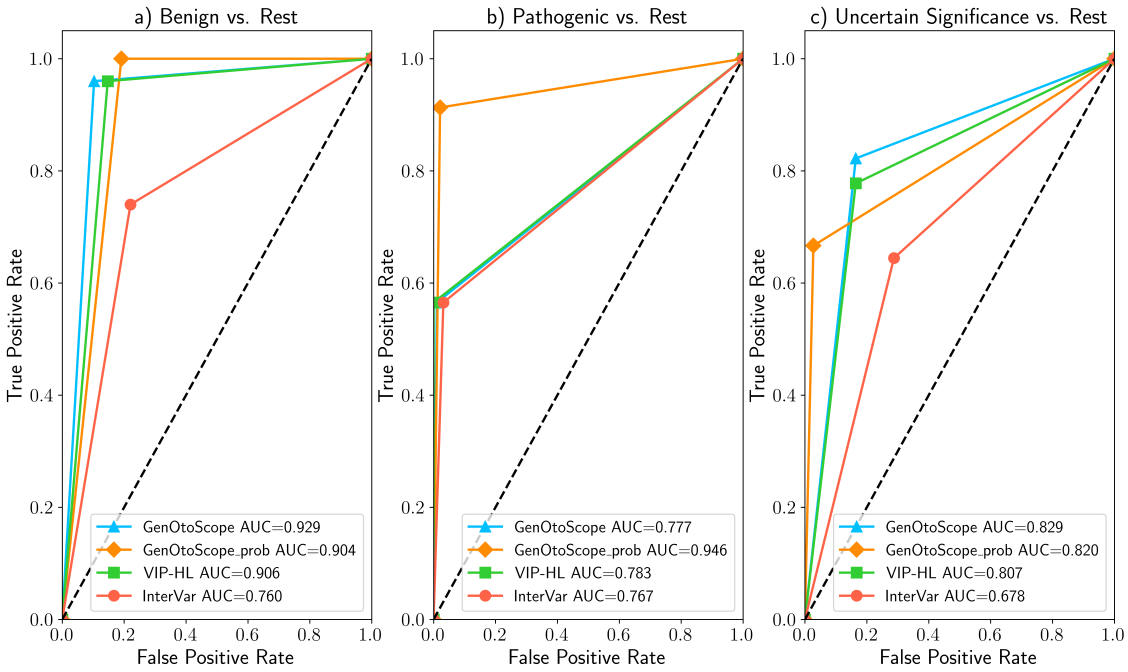


Figure 5.11. ROC curves and AUC scores of all classification tools for MHH data set: a) Prediction of the "Benign" broader class versus the "Pathogenic" broader class and the VUS class b) Prediction of the "Pathogenic" broader class versus "Benign" broader class and the VUS class c) Prediction of the "VUS" class versus the "Benign" broader class and the "Pathogenic" broader class.

ratio of the activation frequency of a given tool compared to the activation frequency of the manual curation in Fig. 5.13.

For the "Pathogenic" broader class, VIP-HL activated 5 evidence-based rules (PVS1 (Strong), PP3, PM2, PS1 and PVS1) from 8 times less (PVS1 (Strong)) to 16 times less (PVS1). In contrast, for the same class, `GenOtoScope` activated 3 out of these 8 rules with the same frequency (PVS1, PS1 and PM2) as the manual curation. The remaining two rules were activated approximately one time more (PP3) and twice more often (PVS1 (Strong)) as the manual curation, respectively.

VIP-HL activated 8 implemented rules (BP7, BP4, BS1 (Supporting), BS1, PM2 (Supporting), PP3, PM5 and PM2) from 20 times less (BP4 and BP7) to 40 times less (PM2) than the manual curation for the "VUS" class. `GenOtoScope` activated 2 of these 8 rules (PM5 and PP3) with equal frequency to the manual curation. `GenOtoScope` activated the remaining six rules (PM2, PM2 (Supporting), BS1, BP4, BS1 (Supporting) and BP7) with approximately one time more (BP7), up to one time less (PM2) as the manual curation.

For the "Benign" broader class, VIP-HL activated 6 rules (PP3, BS1 (Supporting), BP7, BS1, BA1 and BP4) from 12 times less (PP3) to 25 times less (BP4) than the

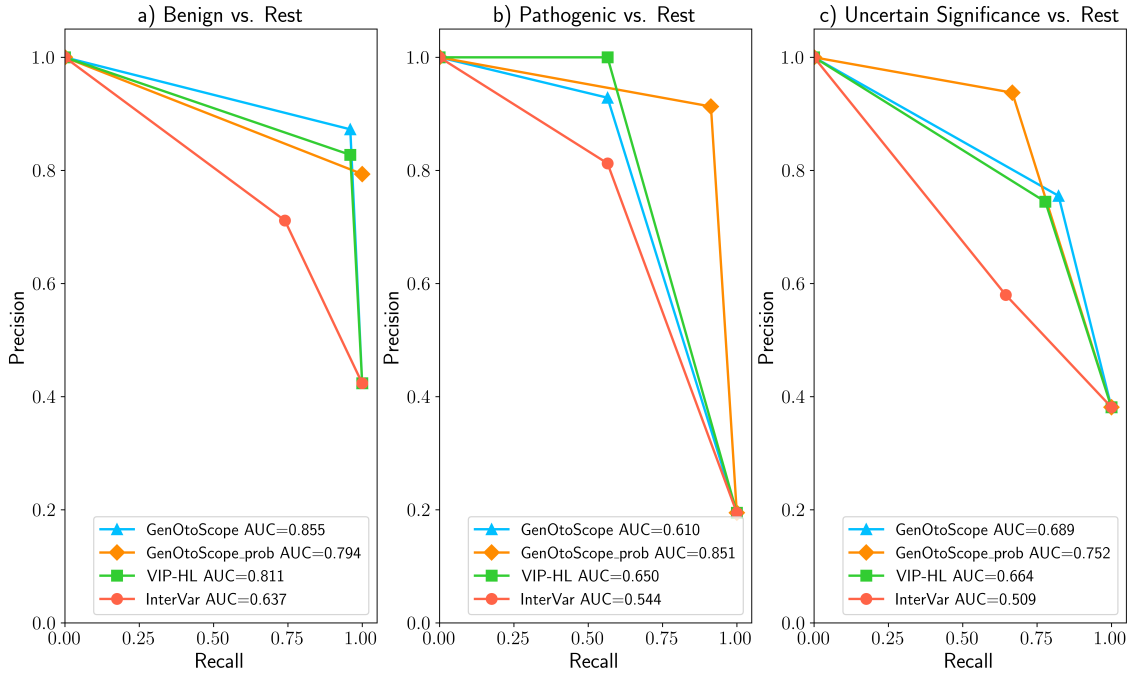


Figure 5.12. Precision-recall curves and AUC scores of all classification tools for the MHH data set: a) Prediction of "Benign" broader class versus the "Pathogenic" broader class and the VUS class b) Prediction of the "Pathogenic" broader class versus "Benign" broader class and the VUS class c) Prediction of the "VUS" class versus the "Benign" broader class and the "Pathogenic" broader class.

manual curation. Contrary to VIP-HL pattern, `GenOtoScope` activated 2 out of these 6 rules (BA1 and BS1) with the same frequency as the manual classification and the remaining 4 rules (BP4, BS1 (Supporting), BP7 and PP3) with approximately two times more (PP3) up to one time less (BP4) as the manual curation.

Based on the observed motives on the activation frequency of each tool compared to the manual curation, we conclude that VIP-HL activated the aforementioned evidence-based rules less frequently than the manual curation. However, `GenOtoScope` was able to trigger the selected rules with similar or at most twice higher frequency compared to the manual curation. Consequently, we justify the best performance achieved in ROC and Precision-Recall scores by `GenOtoScope` for all three broader classes compared to the other two classification tools.

5.3.4 Constructing Annotations for ACMG Criteria

In the following, we explain how `GenOtoScope` utility scripts create the needed annotations for the automatic examination of the ACMG criteria. First, we present our methods to construct the annotations for PVS1 criterion: (i) critical regions for pro-

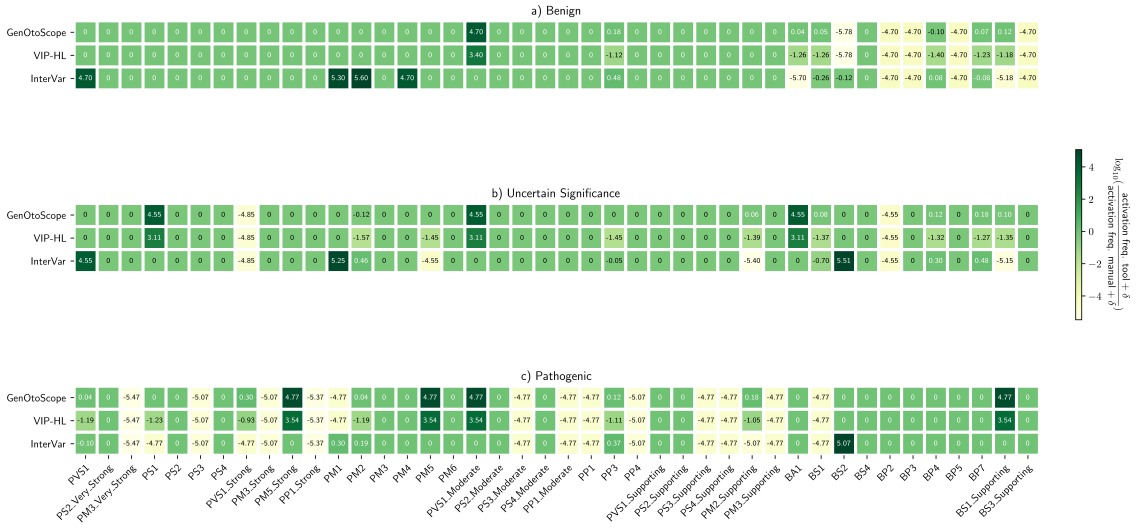


Figure 5.13. Activation frequency ratios for MHH data set. Log ratios calculated for each of the three classes classified by the MHH manual curators: (a) the "Benign" broader class, (b) the "VUS" class and (c) the "Pathogenic" broader class.

tein function, (ii) clinical significant exons and (iii) HL-relevant transcripts. Finally, we present how the respective `GenOtoScope` utility script creates the critical regions for protein function with no benign mutation.

Critical Regions for Protein Function

To automate the assessment of PVS1 rule, the GenOtoScope's sub-process construct three annotation files. In the first annotation file we include all critical regions for protein function. To create this file, the GenOtoScope's respective sub-process maps all available ClinVar entries to the genomic positions of each UniProt domain, respecting the genomic strand of the domain. Then it filters-in all mapped ClinVar entries with at least 2 quality stars on their review status field. For each domain, it uses the interpretation field of the filtered-in overlapping ClinVar entries to calculate the probability that the region is pathogenic (and so critical for protein function) as:

$$\mathrm{P}^{\mathrm{d}}_{\mathrm{pathogenic}} = \frac{N^{d}_{\mathrm{pathogenic}} + \delta}{\sum_{c=\{\mathrm{pathogenic,VUS,benign}\}} (N^{d}_{c} + \delta)}, \tag{5.4}$$

where N^{d}_{c} is the number of filtered ClinVar entries, found in the protein domain with UniProt id d, with class equal to c. The parameter $\delta = 10^{-6}$ is used as a smoothing parameter for the probability computation. Finally, the sub-process calls domains with $\mathrm{P}^{\mathrm{d}}_{\mathrm{pathogenic}} \geq 0.51$ as critical for protein function. It saves all these critical domains for protein function in a BED file containing as columns: their genomic position,

strand, their protein UniProt id and their corresponding $P^{d}_{pathogenic}$ probability. The described procedure is also depicted in the Fig. 5.14.

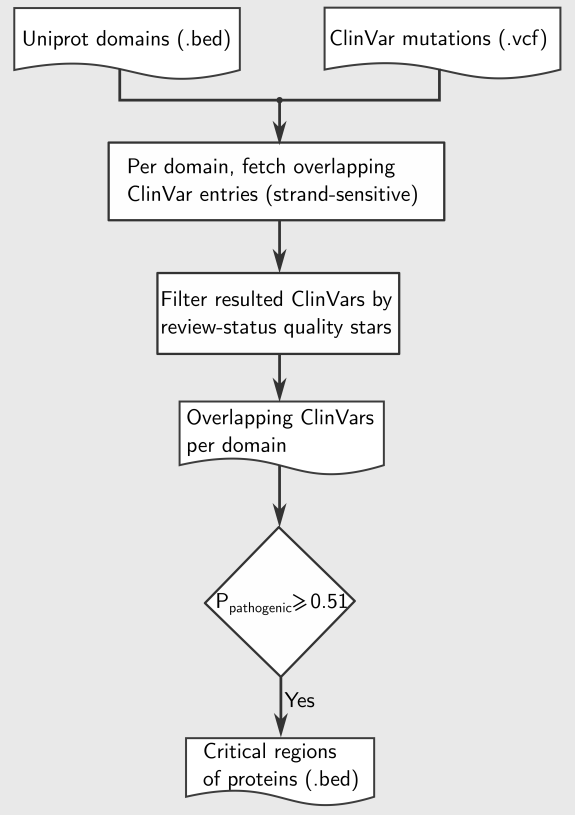


Figure 5.14. Conceptual workflow to call critical regions of proteins for assessment of PVS1 rule.

Clinically Significant Exons

We also developed a sub-process to create an annotation file for clinically significant exons, which are exons at which loss of function variants are not frequent in the general population Abou Tayoun et al. (2018). To do so, the sub-process first aggregates putative loss of functions (pLoF) variants of gnomAD Karczewski et al. (2020) per Ensembl exon Howe et al. (2021). Second, for a given exon, it extracts the AF for each subpopulation of a pLoF variant intersecting the exon. Finally, it aggregates the AF of each extracted pLoF variant, for each subpopulation, and if this sum is lower than 0.001 for any subpopulation, the exon is called clinically significant. The output annotation file is in BED format containing the columns: the genomic position, strand, its exon Ensembl id and all containing transcript Ensembl ids, for each called clinical significant exon. The procedure is depicted in Fig. 5.15.

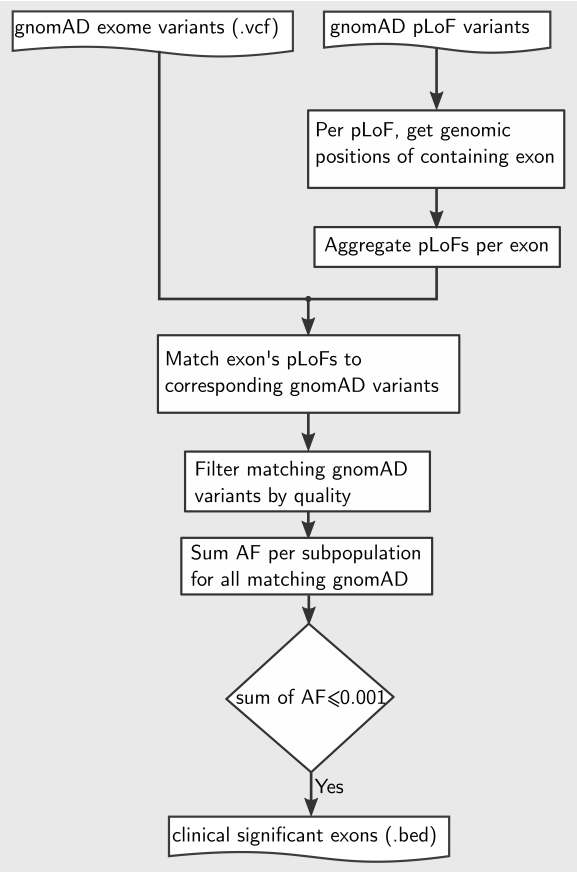


Figure 5.15. Conceptual workflow to call clinical significant exons for PVS1 rule.

Hearing Loss-relevant Transcripts

The last annotation file for PVS1 is the hearing loss relevant transcripts. The respective sub-process utilizes three independent annotation files to create the hearing loss relevant transcripts and exons. The first file contains the phenotype relevant transcripts and their clinically relevant exons from DiStefano et al. (2018), the second file contains disease-gene pairs for hearing loss from ClinGen repository DiStefano et al. (2019) and the last file contains the clinical diagnostics panel for hearing loss created by the HG department of the MHH.

To aggregate these files, we argue that the DiStefano et al. (2018) work contains the most detailed information as it contains not only hearing loss relevant transcripts for a given gene but also the clinically relevant exons of the respective transcripts. In contrast, the two remaining files contain either disease-gene pairs or only relevant transcripts of genes without specifying their clinically relevant exons. Therefore, the sub-process extends the relevant transcripts and clinically relevant exons reported by DiStefano et al. (2018) with the relevant transcripts and all their contained exons from the diagnostic panel of HG department of MHH. Further, it extracts the longest

coding transcript or the clinical relevant transcript annotated by Locus Reference Genomic resource (LRG) MacArthur et al. (2014), and their contained transcripts, for all genes annotated by ClinGen but not found in the extended list of annotations of the last intermediate step. As final step, it aggregates the ClinGen unique genes annotations with DiStefano et al. (2018) and the HG department of the MHH annotations to create the final hearing loss relevant transcripts and clinically relevant exons. The output annotation file is in BED format, containing the columns: chromosomal position, strand, transcript and exon ids for each clinically relevant exon.

Critical Regions for Protein Function With No Benign Mutation

To automate PM1 evidence-based rule we need an annotation file containing the critical regions for protein function without benign mutations. To construct these annotated regions, we implemented a similar sub-process as for the critical regions used for PVS1 rule. The only difference is that this sub-process constraints the candidate domains with $P_{pathogenic} \leq 0.51$ (Eq.5.4) to contain no benign ClinVar mutations. The resulting file is in BED format, containing the same columns as described above for the critical regions for protein function.

PVS1 Annotations

`GenOtoScope` sub-processes created the three annotation files needed for the refined PVS1 criterion. For the first file, critical regions, we applied our methodology using the 25,552 ClinVar entries, version of March 2021 and 12,776 UniProt domains, version of February 2021. The resulting file contains 1,478 UniProt domains annotated as critical for protein function. Using the HL-relevant transcripts and exons curated in DiStefano et al. (2018), DiStefano et al. (2019) (VCEP-HL) and MHH diagnostic panel, we extracted 2,812 unique exons and 215 unique transcripts, contained in 154 genes. For the annotation file with the clinical significant exons, the used version of the pLoF variants and the allele frequency of exomes of gnomAD was the version 2.1.1. By this process, we annotated 107,966 exons as clinical significant exons.

PM1 Annotations

The 10 mutational hotspots relevant to HL as published by VCEP-HL committee, at page[5], were utilized to create the annotation file to evaluate the PM1 criterion. Using the ClinVar entries and UniProt domains, same versions as described above. Besides 750 UniProt domains were called as critical regions for protein function without a benign variant. To evaluate the PM1 evidence-criteria we intersect the chromosomal position of the input variant with both annotation files.

[5] https://submit.ncbi.nlm.nih.gov/ft/byid/vroiax8b/hearing_loss_acmg_specifications_v1_2018.pdf

5.3.5 Disclaimer

The classification produced by `GenOtoScope` is intended for an efficient pathogenicity prediction of WES files, thus for research use only. It is not intended for diagnostic or clinical purposes. The classification provided by `GenOtoScope` does not replace a physician's medical judgment and usage is entirely at your own risk. The providers of this resource shall in no event be liable for any direct, indirect, incidental, consequential, or exemplary damages.

5.4 Conclusion

In this chapter, we presented `GenOtoScope`, an automated classification tool for variants associated with congenital HL. By this work we answered towards the third research questions of thesis.

We have shown that `GenOtoScope` outperformed other variant classification tools in terms of AUC score of ROC curve and of Precision-recall curve for all three broader classes ("Benign", "VUS" and "Pathogenic"). To explain the difference in performance between the tools, we calculated the ratio of the activation frequency of triggered criteria by each tool and the manual curation. By comparing the ratios for each ACMG criterion, we observed that `GenOtoScope` achieved the most similar activation frequency to the manual curation, compared to the VIP-HL and InterVar.

Besides, the scope of this work is to provide an easily accessible tool to use for the classification of variants for HL phenotype. Therefore, we implemented two versions of the tool for two different scenarios. A CLI version to be used by experienced bioinformatics personnel aiming to classify a set of patients WES VCF files and a web interface to be used by other life scientists, with no bioinformatics expertise, to classify a single variant of interest. We hope that this tool will be applied in research settings of molecular genetics to provide a time-efficient and standardized classification of HL variants.

For future extension of `GenOtoScope` we aim to implement the most frequently activated evidence-based criteria by manual curation to predict the two complementary broader classes. For the "Benign" broader class, the not implemented criteria with highest activation frequency, by the manual curation, were the BS2, BP2, BP3, BP5 and BS3 (Supporting). For the "Pathogenic" broader class, the most frequently activated criteria, by the manual curation, were PM3, aggregated for all strengths, PP1, aggregated for all strengths, PS3 and PS4. To implement these criteria which need heavily manual curation, we aim to utilize databases for genotype to phenotype such as DisGeNET Piñero et al. (2020) or to use prediction algorithms to link a mutation of interest to its respective functional study publications, for example AVADA Birgmeier et al. (2020). Last, methods should be implemented to automatically examine the segregation criteria, whenever genomic data of the patient's family

are available. Also, to facilitate even more the personnel with no bioinformatics experience to use the web interface, we would allow the user to input a single variant information, without the need of creating a VCF file on the `GenOtoScope` website.

Last, by making the GenOtoScope an open source project, we aim to facilitate researchers to expand its range of usage to other phenotypes compatible with ACMG-based analysis - for example cardiomyopathy Kelly et al. (2018) or monogenic diabetes[6] – by adjusting specific thresholds, providing customized annotation files and adapting the source code if needed.

[6] https://clinicalgenome.org/site/assets/files/7039/clingen_diabetes_acmg_specifications_v1.pdf

6 Conclusions and Future Work

6.1 Conclusion and Contributions

Artificial Intelligence (AI) aims to design and develop practical solutions to automate everyday or industrial processes. It spans a wide range of applications from email spam detection to self-automating vehicles. An important step of AI applications is to categorize input instances into a predefined set of classes. For example, the image detection system on the car-assembly pipeline, needs to recognize the compartment of the car, front, middle or rear. This step is known as classification. The classification is significant as it enables the AI system to perform a set of similar (processing) steps for input of the same class. That is, in the previous example, AI system can trigger the mechanical arm to perform the same processing steps for each front compartment of a car. Methods to perform the classification steps can be based on domain experts rules or data-driven methods. All classification methods are composed of the four steps: (i) input representation, (ii) model building (iii) model prediction and (iv) model assessment.

Based on the input data characteristics, the availability of labeled data and the need for model assessment by experts, there many challenges on applying the classification steps. That is, in Chapter 3, I discussed my work on improving the model building step to tackle the concept drift phenomenon on classifying the textual streams. In Chapter 4, I improved the input representation step by learning neural representation for protein domains which can be intrinsically assessable and improved the performance for protein learning tasks. In the last chapter 5, I described my method to classify human genomic variants associated with hearing loss based on well-accepted criteria, created by ACMG/AMP Oza et al. (2018). Therefore, I contributed on a classification model that enables the model assessment step for human variants pathogenicity prediction.

6.1.1 Contributions

Our contributions are summarized as follows:

Classifying Textual Streams under Feature Drifts

In Chapter 3, we defined feature drift phenomenon as a special type of concept drift. To tackle feature drifts in learning over a stream of documents I made the three contributions. First, I conceptualized four complementary temporal processes that may capture the periodical pattern of a word of a document. Then, I aggregated the predictions of these four processes, by an ensemble learning model, using the "sequential prediction with expert advice" framework, Kivinen and Warmuth (1999), to guarantee a minimal prediction loss compared to the process best explaining the word periodicity. Our second contribution was to deploy these predictions, for each word, in order to enhance MNB model so that it can predict document class, in an online fashion, under feature drifts. I deployed a sketch algorithm to keep the most frequent used words for each period of the stream and thus I restrict the main memory resources and allow our model, `temporalMNB`, to predict in "anytime protocol". More specifically `temporalMNB` needs 0.0253 seconds for both training and predicting of a new instance and the needs 20GB at maximum for a tested Twitter stream of 1.6 million tweets. Our third contribution was to compare `temporalMNB` to other time-aware MNB approaches in three tasks, sentiment prediction in a Twitter stream of 1.6 million tweets, email spam detection for two smaller data sets. Our results show that `temporalMNB` outperformed the other approaches for all three tasks. Last, I verified the prediction guarantees for words with known periodicities from the two spam detection data sets.

Classifying Protein Sequences using Intrinsically Assessable Domain Embeddings

In Chapter 4, I presented our method to improve input representation for full-protein learning tasks. This work has three contributions. First, I applied the `word2vec` algorithms to learn embeddings for protein domains, named `dom2vec`, based on domain architectures provided by the InterPro database Blum et al. (2021). Importantly, as metadata for the most important biological characteristics exist, our protein domains embeddings can be intrinsically evaluated in a quantitative manner. Thus, compared to amino acid sequence embeddings, our approach brings the advantage of enabling quantitative validation of the captured knowledge by the embeddings, before using them on downstream tasks. Second, I performed quantitative evaluation for four biological characteristics of proteins domains: domain hierarchy, secondary structure class, enzymatic commission class and Gene Ontology molecular function class for three model organisms and a human pathogen, (*Escherichia coli*, *Saccharomyces cerevisiae*, *Homo sapiens* and *Plasmodium falciparum*). Our experiments

showed that `dom2vec` embeddings captured the most important biological features of an individual domain: secondary structure class, enzymatic and molecular function classes. Therefore, our findings supported, in a data-driven approach, the accepted modular evolution of proteins Moore et al. (2008). Besides, they proved that neural networks can capture language information for the approved grammar of protein domains Yu et al. (2019), in a full parallel to the captured semantic and lexical features by neural word embeddings in natural languages. Our last contribution was to benchmark the performance gains for three protein learning tasks compared to sequence embeddings. I showed that `dom2vec` outperformed state-of-the-art sequence embedings for the prediction tasks of enzymatic commission class and toxicity presence. Last, I investigated the performance of `dom2vec` for a data set of which the test set contains unseen domains, compared to the training set. For such out-of-vocabulary scenario, I showed that if the unknown domains are from the range of 0-30% `dom2vec` embeddings were comparable with sequence embeddings.

Classifying Human Genomic Variants Associated with Congenital Hearing Loss

In Chapter 5, I discuss my contribution to improve model assessment step on the scenario of classifying the pathogenicity of human genomic variants. I designed and developed a open source bioinformatics tool, *GenOtoScope*, to classify the pathogenicity of variants associated with congenital hearing loss based on the ACMG/AMP guidelines. Our second contribution was to compare *GenOtoScope* performance to two other classification tools on two hearing loss variants data sets. I showed that *GenOtoScope* outperformed both classification tools. Importantly, I verified the high performance of *GenOtoScope*, by observing that our tool trigger the ACMG/AMP evidence-based criteria in the most similar fashion, to the manual curators, as compared to the other two classification tools. Our final contribution was to develop two software interfaces for *GenOtoScope*, a command line and a website, enable the wide, free and easy usage of our tool by the research community and medical diagnostics personnel.

6.2 Future Work Directions

The ideas on future work directions are split into the three main topics of this thesis. On online learning algorithms for drifting textual streams, I achieved to enhance MNB model, however a more complex model can capture better the expressiveness of textual documents. Therefore, a research direction is to apply the "sequential prediction with expert advice" framework for more complex models such as recurrent neural networks (RNNs) Graves and Schmidhuber (2008). To do so, researchers can conceptualize an ensemble model which has RNNs of different time periods as

experts optimally combining by expert advice framework. As I proved that expert advice framework can provide practical guarantees for the expert prediction aggregation, such ensemble of RNNs may acquire a high performance on classifying a textual stream over concept drifts.

Next, I describe two future research ideas for `dom2vec` embeddings. The first research direction is to use `dom2vec` embeddings to improve the functional prediction of domains of unknown function (DUF). That is, assuming that DUFs are similar to rare words that a human reads in a large collection of corpora, I will also apply embedding models for tiny data, such as *nonce2vec* Herbelot and Baroni (2017), to improve the captured information for DUF vectors. These re-trained DUF vectors, will be input to a learning model to predict the domain function. Our second research direction is to use `dom2vec`embeddings for functional characterization of hypothetical proteins; proteins with no alignment to a protein with a known function. For this direction, the embedding of the whole protein architecture can be learned and then sequence embeddings can be trained through a siamese neural network, Chicco (2021), with input the domain architectures and amino acid sequences of a pair of proteins. Finally, the representation of hypothetical protein can be retrieved from the trained siaseme network and it can be contrasted to protein with known function.

Last, I describe research ideas based on the *GenOtoScope* bioinformatics tool. First, research can be conducted to increase the coverage of automatic examination of even more ACMG/AMP evidence-based criteria. To do so, new methods needed to be developed to connect, automatically, a human variant signature to the published works with functional studies for this variant. Second, researchers can take advantage of the open source license of GenOtoScope and use its code as code base to implement the ACMG/AMP evidence-based criteria for more disease phenotype such as cancer or hereditary cardiomyopathy.

A

Curriculum Vitae

Academic Experience

since 2017 **PhD candidate in Computer Science** in L3S Research Center at Gottfried Wilhelm Leibniz University, Germany

2012–2014 **Master of Science in Computational Biology and Bioinformatics** at Swiss Federal Institute of Technology Zürich (ETH) and University of Zürich, Switzerland

2006–2011 **Engineering Diploma (5-year studies) in Computer Science** at Computer Engineering & Informatics Department, University of Patras, Greece

Working Experience

since 2022 **Bioinformatics Scientist** at Platomics GmbH, Vienna, Austria

since 2020 **Project Member** of Cochlear Implant (CI) project (funded by Volkswagen Foundation, VolkswagenStiftung)

2020–2021 **Project Member** of BacData project (funded by Volkswagen Foundation, VolkswagenStiftung)

2017–2018 **Project Member** of OSCAR project (funded by Federal Ministry of Education and Research, BMBF)

2017–2018 **Teaching Assistant** for Data Mining I and Data Mining II courses

2014–2015 **Research Assistant (Internship) position** at the research group of Molecular Diagnostics, Genomics and Bioinformatics at Agroscope Swiss Confederation's center of excellence for agricultural research, Zürich, Switzerland

---

Publications

Please refer to the foreword of this thesis for a full list of my publications.

Bibliography

Abou Tayoun, A. N., Pesaran, T., DiStefano, M. T., Oza, A., Rehm, H. L., Biesecker, L. G., Harrison, S. M., and SVI), C. S. V. I. W. G. C. (2018). Recommendations for interpreting the loss of function pvs1 acmg/amp variant criterion. *Human mutation*, 39(11):1517–1524.

Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G. M. (2019). Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322.

Almeida, F. and Xexéo, G. (2019). Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*.

Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., Iotti, E., Magliani, F., and Manicardi, S. (2016). A comparison between preprocessing techniques for sentiment analysis in Twitter. In *KDWeb*.

Asgari, E., McHardy, A. C., and Mofrad, M. R. K. (2019). Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). *Scientific Reports*, 9(3577).

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Attardi, G., Cozza, V., and Sartiano, D. (2015). Detecting the scope of negations in clinical notes. *CLiC it*, page 14.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16.

Barddal, J. P., Gomes, H. M., Enembreck, F., and Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, 127:278–294.

Bedbrook, C. N., Yang, K. K., Rice, A. J., Gradinaru, V., and Arnold, F. H. (2017). Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS computational biology*, 13(10):e1005786.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Bepler, T. and Berger, B. (2019). Learning protein sequence embeddings using information from structure. In *Proceedings of the* 7^{th} *International Conference on Learning Representations*.

Berrios, C., Hurley, E. A., Willig, L., Thiffault, I., Saunders, C., Pastinen, T., Goggin, K., and Farrow, E. (2021). Challenges in genetic testing: clinician variant interpretation processes and the impact on clinical care. *Genetics in Medicine*, pages 1–11.

Bifet, A. and Frank, E. (2010). Sentiment knowledge discovery in Twitter streaming data. In *International conference on discovery science*, pages 1–15. Springer.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

Bifet, A., Holmes, G., and Pfahringer, B. (2011). MOA-tweetreader: real-time analysis in Twitter streaming data. In *International Conference on Discovery Science*, pages 46–60. Springer.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM.

Birgmeier, J., Deisseroth, C. A., Hayward, L. E., Galhardo, L. M., Tierno, A. P., Jagadeesh, K. A., Stenson, P. D., Cooper, D. N., Bernstein, J. A., Haeussler,

M., et al. (2020). Avada: toward automated pathogenic variant evidence retrieval directly from the full-text literature. *Genetics in Medicine*, 22(2):362–370.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blum, M., Chang, H.-Y., Chuguransky, S., Grego, T., Kandasaamy, S., Mitchell, A., Nuka, G., Paysan-Lafosse, T., Qureshi, M., Raj, S., et al. (2021). The interpro protein families and domains database: 20 years on. *Nucleic acids research*, 49(D1):D344–D354.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Box, G. E. and Jenkins, G. M. (1976). *Time series analysis: forecasting and control, revised ed*. Holden-Day.

Bu, F., Zhong, M., Chen, Q., Wang, Y., Zhao, X., Zhang, Q., Li, X., Booth, K. T., Azaiez, H., Lu, Y., et al. (2022). Dvpred: a disease-specific prediction tool for variant pathogenicity classification for hearing loss. *Human Genetics*, 141(3):401–411.

Buchan, D. W. and Jones, D. T. (2020). Learning a functional grammar of protein domains using natural language word embedding techniques. *Proteins: Structure, Function, and Bioinformatics*, 88(4):616–624.

Canakoglu, A., Pinoli, P., Bernasconi, A., Alfonsi, T., Melidis, D. P., and Ceri, S. (2021). Virusurf: an integrated database to investigate viral sequences. *Nucleic acids research*, 49(D1):D817–D824.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Chicco, D. (2021). Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94.

Chou, K.-C. and Cai, Y.-D. (2002). Using functional domain composition and support vector machines for prediction of protein subcellular location. *Journal of Biological Chemistry*, 277(48):45765–45769.

Chunn, L. M., Nefcy, D. C., Scouten, R. W., Tarpey, R. P., Chauhan, G., Lim, M. S., Elenitoba-Johnson, K. S., Schwartz, S. A., and Kiel, M. J. (2020). Mastermind: a comprehensive genomic association search engine for empirical evidence curation and genetic variant interpretation. *Frontiers in Genetics*, 11.

Clarke, L., Fairley, S., Zheng-Bradley, X., Streeter, I., Perry, E., Lowy, E., Tassé, A.-M., and Flicek, P. (2017). The international genome sample resource (igsr): A worldwide collection of genome variation incorporating the 1000 genomes project data. *Nucleic acids research*, 45(D1):D854–D859.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423.

Collins, F. S., Morgan, M., and Patrinos, A. (2003). The human genome project: lessons from large-scale biology. *Science*, 300(5617):286–290.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Consortium, U. (2021). Uniprot: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489.

Cooper, G. M., Stone, E. A., Asimenos, G., Green, E. D., Batzoglou, S., and Sidow, A. (2005). Distribution and intensity of constraint in mammalian genomic sequence. *Genome research*, 15(7):901–913.

Cryer, J. and Chan, K. (2008). Time series analysis with applications in R.

Dale, R. K., Pedersen, B. S., and Quinlan, A. R. (2011). Pybedtools: a flexible python library for manipulating genomic datasets and annotations. *Bioinformatics*, 27(24):3423–3424.

Delany, S. J., Cunningham, P., Tsymbal, A., and Coyle, L. (2004). A case-based technique for tracking concept drift in spam filtering. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 3–16. Springer.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

DiStefano, M. T., Hemphill, S. E., Cushman, B. J., Bowser, M. J., Hynes, E., Grant, A. R., Siegert, R. K., Oza, A. M., Gonzalez, M. A., Amr, S. S., et al. (2018). Curating clinically relevant transcripts for the interpretation of sequence variants. *The Journal of Molecular Diagnostics*, 20(6):789–801.

DiStefano, M. T., Hemphill, S. E., Oza, A. M., Siegert, R. K., Grant, A. R., Hughes, M. Y., Cushman, B. J., Azaiez, H., Booth, K. T., Chapin, A., et al. (2019). Clingen expert clinical validity curation of 164 hearing loss gene–disease pairs. *Genetics in Medicine*, 21(10):2239–2247.

Doğan, T., MacDougall, A., Saidi, R., Poggioli, D., Bateman, A., O'Donovan, C., and Martin, M. J. (2016). UniProt-DAAC: domain architecture alignment and classification, a new method for automatic functional annotation in UniProtKB. *Bioinformatics*, 32(15):2264–2271.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80.

Drozd, A., Gladkova, A., and Matsuoka, S. (2016). Word embeddings, analogies, and machine learning: Beyond king-man+woman=queen. In *Proceedings of the* 26^{th} *International Conference on Computational Linguistics: Technical Papers*.

Duda, R. O., Hart, P. E., et al. (1973). *Pattern classification and scene analysis*, volume 3. Wiley New York.

Emanuelsson, O., Nielsen, H., Brunak, S., and Von Heijne, G. (2000). Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of Molecular Biology*, 300(4):1005–1016.

Fleischmann, A., Darsow, M., Degtyarenko, K., Fleischmann, W., Boyce, S., Axelsen, K. B., Bairoch, A., Schomburg, D., Tipton, K. F., and Apweiler, R. (2004). Intenz, the integrated relational enzyme database. *Nucleic acids research*, 32(suppl_1):D434–D437.

Forslund, K. and Sonnhammer, E. L. (2008). Predicting protein function from domain content. *Bioinformatics*, 24(15):1681–1687.

Forslund, K. and Sonnhammer, E. L. (2012). Evolution of protein domain architectures. *Evolutionary Genomics*, pages 187–216.

Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2013). SCOPe: Structural classification of proteins-extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309.

Fox, R. J., Davis, S. C., Mundorff, E. C., Newman, L. M., Gavrilovic, V., Ma, S. K., Chung, L. M., Ching, C., Tam, S., Muley, S., et al. (2007). Improving catalytic function by prosar-driven enzyme evolution. *Nature biotechnology*, 25(3):338–344.

Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction*. springer open.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2):131–163.

Gacesa, R., Barlow, D. J., and Long, P. F. (2016). Machine learning can differentiate venom toxins from other proteins having non-toxic physiological functions. *PeerJ Computer Science*, 2:e90.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

Giachanou, A. and Crestani, F. (2016). Tracking sentiment by time series analysis. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1037–1040. ACM.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Gomes, H. M., Barddal, J. P., Enembreck, F., and Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):1–36.

Graves, A. and Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*, 21:545–552.

Hamosh, A., Scott, A. F., Amberger, J., Valle, D., and McKusick, V. A. (2000). Online mendelian inheritance in man (omim). *Human mutation*, 15(1):57–61.

Hand, D. J. and Yu, K. (2001). Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Heinzinger, M., Elnaggar, A., Wang, Y., Dallago, C., Nechaev, D., Matthes, F., and Rost, B. (2019). Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, 20(723).

Herbelot, A. and Baroni, M. (2017). High-risk learning: acquiring new word vectors from tiny data. *arXiv preprint arXiv:1707.06556*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer.

Hoens, T. R., Polikar, R., and Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101.

Hofmeister, F. (1890). Ueber die darstellung von krystallisirtem eieralbumin und die krystallisirbarkeit colloider stoffe. *Zeitschrift für Physiologische Chemie*, 14:165–172.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10.

Howe, K. L., Achuthan, P., Allen, J., Allen, J., Alvarez-Jarreta, J., Amode, M. R., Armean, I. M., Azov, A. G., Bennett, R., Bhai, J., et al. (2021). Ensembl 2021. *Nucleic acids research*, 49(D1):D884–D891.

Hu, Z., Yau, C., and Ahmed, A. A. (2017). A pan-cancer genome-wide analysis reveals tumour dependencies by induction of nonsense-mediated decay. *Nature communications*, 8(1):1–9.

Huang, S., Zhao, G., Wu, J., Li, K., Wang, Q., Fu, Y., Zhang, H., Bi, Q., Li, X., Wang, W., et al. (2021). Gene4hl: An integrated genetic database for hearing loss. *Frontiers in genetics*, page 2009.

Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106.

Hunt, E. B., Marin, J., and Stone, P. J. (1966). Experiments in induction. *The Computer Journak*, 10:299–299.

Ioannidis, N. M., Rothstein, J. H., Pejaver, V., Middha, S., McDonnell, S. K., Baheti, S., Musolf, A., Li, Q., Holzinger, E., Karyadi, D., et al. (2016). Revel: an ensemble method for predicting the pathogenicity of rare missense variants. *The American Journal of Human Genetics*, 99(4):877–885.

Jian, X., Boerwinkle, E., and Liu, X. (2014). In silico prediction of splice-altering single nucleotide variants in the human genome. *Nucleic acids research*, 42(22):13534–13544.

Jianqiang, Z. and Xiaolin, G. (2017). Comparison research on text pre-processing methods on Twitter sentiment analysis. *IEEE Access*, 5:2870–2879.

Jones, P. and *et al.* (2014). InterProScan 5: genome-scale protein function classification. *Bioinformatics*, 30(9):1236–1240.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the* 15^{th} *Conference of the European Chapter of the Association for Computational Linguistics*.

Karczewski, K. J., Francioli, L. C., Tiao, G., Cummings, B. B., Alföldi, J., Wang, Q., Collins, R. L., Laricchia, K. M., Ganna, A., Birnbaum, D. P., et al. (2020). The mutational constraint spectrum quantified from variation in 141,456 humans. *Nature*, 581(7809):434–443.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2006). Dynamic feature space and incremental feature selection for the classification of textual data streams. *Knowledge Discovery from Data Streams*, pages 107–116.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 22(3):371–391.

Kelly, M. A., Caleshu, C., Morales, A., Buchan, J., Wolf, Z., Harrison, S. M., Cook, S., Dillon, M. W., Garcia, J., Haverfield, E., et al. (2018). Adaptation and validation of the acmg/amp variant classification framework for myh7-associated inherited cardiomyopathies: recommendations by clingen's inherited cardiomyopathy expert panel. *Genetics in Medicine*, 20(3):351–359.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the* 3^{rd} *International Conference on Learning Representations*.

Kivinen, J. and Warmuth, M. K. (1999). Averaging expert predictions. In *European Conference on Computational Learning Theory*, pages 153–167. Springer.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent data analysis*, 8(3):281–300.

Kopanos, C., Tsiolkas, V., Kouris, A., Chapple, C. E., Aguilera, M. A., Meyer, R., and Massouras, A. (2019). Varsome: the human genomic variant search engine. *Bioinformatics*, 35(11):1978.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Lai, C., Zimmer, A. D., O'Connor, R., Kim, S., Chan, R., van den Akker, J., Zhou, A. Y., Topper, S., and Mishne, G. (2020). Leap: Using machine learning to support variant classification in a clinical setting. *Human mutation*, 41(6):1079–1090.

Landgraf, C., Melidis, D. P., Schmidt, G., Schöner-Heinisch, A., von Hardenberg, S., Auber, B., and Nejdl, W. (2021). Genotoscope: Automated annotation of variants associated with hereditary hearing loss. In *European Human Genetics Conference*.

Landgraf, C., Melidis, D. P., Schmidt, G., Schöner-Heinisch, A., von Hardenberg, S., Förster, A., Lesinski-Schiedat, A., Auber, B., and Nejdl, W. (2022a). Genotoscope: Automated annotation of variants associated with hereditary hearing loss. In *German Society of Human Genetics*.

Landgraf, C., Melidis, D. P., Schmidt, G., Schöner-Heinisch, A., von Hardenberg, S., Katzke, A.-L., Förster, A., Lesinski-Schiedat, A., Auber, B., and Nejdl, W. (2022b). Genotoscope: Towards automated acmg classification of genetic variants associated with congenital hearing loss. In *European Human Genetics Conference*.

Landrum, M. J., Lee, J. M., Benson, M., Brown, G. R., Chao, C., Chitipiralla, S., Gu, B., Hart, J., Hoffman, D., Jang, W., et al. (2018). Clinvar: improving access to variant interpretations and supporting evidence. *Nucleic acids research*, 46(D1):D1062–D1067.

Lastra-Díaz, J. J., Goikoetxea, J., Taieb, M. A. H., García-Serrano, A., Aouicha, M. B., and Agirre, E. (2019). A reproducible survey on word embeddings and ontology-based methods for word similarity: linear combinations outperform the state of the art. *Engineering Applications of Artificial Intelligence*, 85:645–665.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Lebanon, G. and Zhao, Y. (2008). Local likelihood modeling of temporal text streams. In *Proceedings of the 25th international conference on Machine learning*, pages 552–559. ACM.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K., Seifert, B. A., Shimelis, H., Ghosh, R., Crowley, S. B., Carter, N. J., Doonanco, K., Foreman, A. K., Ritter, D. I., Jimenez, S., et al. (2019). Clinical validity assessment of genes frequently tested on hereditary breast and ovarian cancer susceptibility sequencing panels. *Genetics in Medicine*, 21(7):1497–1506.

Leondes, C. T. (2001). *Expert systems: the technology of knowledge management and decision making for the 21st century*. Elsevier.

Li, Q. and Wang, K. (2017). Intervar: clinical interpretation of genetic variants by the 2015 acmg-amp guidelines. *The American Journal of Human Genetics*, 100(2):267–280.

Li, Y., Wang, S., Umarov, R., Xie, B., Fan, M., Li, L., and Gao, X. (2017). DEEPre: sequence-based enzyme EC number prediction by deep learning. *Bioinformatics*, 34(5):760–769.

Lin, G., Sun, N., Nepal, S., Zhang, J., Xiang, Y., and Hassan, H. (2017). Statistical twitter spam detection demystified: Performance, stability and scalability. *IEEE access*, 5:11142–11154.

Linder, J. E., Bastarache, L., Hughey, J. J., and Peterson, J. F. (2021). The role of electronic health records in advancing genomic medicine. *Annual Review of Genomics and Human Genetics*, 22.

Loewenstein, Y., Raimondo, D., Redfern, O. C., Watson, J., Frishman, D., Linial, M., Orengo, C., Thornton, J., and Tramontano, A. (2009). Protein function annotation by homology-based inference. *Genome biology*, 10(2):1–8.

Lu, A. X., Zhang, H., Ghassemi, M., and Moses, A. M. (2020). Self-supervised contrastive learning of protein representations by mutual information maximization. *BioRxiv*.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *Proceedings of the* 53^{rd} *Annual Meeting of the Association for Computational Linguistics*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

MacArthur, J. A., Morales, J., Tully, R. E., Astashyn, A., Gil, L., Bruford, E. A., Larsson, P., Flicek, P., Dalgleish, R., Maglott, D. R., et al. (2014). Locus reference genomic: reference sequences for the reporting of clinically relevant sequence variants. *Nucleic acids research*, 42(D1):D873–D878.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Manzoor, E., Lamba, H., and Akoglu, L. (2018). xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1963–1972.

Marchler-Bauer, A., Bo, Y., Han, L., He, J., Lanczycki, C. J., Lu, S., Chitsaz, F., Derbyshire, M. K., Geer, R. C., Gonzales, N. R., et al. (2016). CDD/SPARCLE: functional classification of proteins via subfamily domain architectures. *Nucleic Acids Research*, 45(D1):D200–D203.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

McLaren, W., Gil, L., Hunt, S. E., Riat, H. S., Ritchie, G. R., Thormann, A., Flicek, P., and Cunningham, F. (2016). The ensembl variant effect predictor. *Genome biology*, 17(1):1–14.

Melidis, D. P., Campero, A. V., Iosifidis, V., Ntoutsi, E., and Spiliopoulou, M. (2018a). Enriching lexicons with ephemeral words for sentiment analysis in social streams. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, page 38. ACM.

Melidis, D. P., Landgraf, C., Schmidt, G., Schöner-Heinisch, A., von Hardenberg, S., Lesinski-Schiedat, A., Nejdl, W., and Auber, B. (2022). Genotoscope: Towards automating acmg classification of variants associated with congenital hearing loss. *PLOS Computational Biology*, 18(9):e1009785.

Melidis, D. P. and Nejdl, W. (2021). Capturing protein domain structure and function using self-supervision on domain architectures. *Algorithms*, 14(1):28.

Melidis, D. P., Spiliopoulou, M., and Ntoutsi, E. (2018b). Learning under feature drifts in textual streams. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 527–536.

Metwally, A., Agrawal, D., and El Abbadi, A. (2005). Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, pages 398–412. Springer.

Miikkulainen, R. and Dyer, M. G. (1991). Natural language processing with modular pdp networks and distributed lexicon. *Cognitive Science*, 15(3):343–399.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Mitchell, A. L. and *et al.* (2019). InterPro in 2019: improving coverage, classification and access to protein sequence annotations. *Nucleic Acids Research*, 47(D1):D351–D360.

Moore, A. D., Björklund, Å. K., Ekman, D., Bornberg-Bauer, E., and Elofsson, A. (2008). Arrangements in the modular evolution of proteins. *Trends in biochemical sciences*, 33(9):444–451.

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013). Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402.

Nguyen, H.-L., Woon, Y.-K., Ng, W.-K., and Wan, L. (2012a). Heterogeneous ensemble for feature drifts in data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–12. Springer.

Nguyen, L. T., Wu, P., Chan, W., Peng, W., and Zhang, Y. (2012b). Predicting collective sentiment dynamics from time-series social media. In *Proceedings of the first international workshop on issues of sentiment discovery and opinion mining*, page 6. ACM.

Nicora, G., Limongelli, I., Gambelli, P., Memmi, M., Malovini, A., Mazzanti, A., Napolitano, C., Priori, S., and Bellazzi, R. (2018). Cardiovai: an automatic implementation of acmg-amp variant interpretation guidelines in the diagnosis of cardiovascular diseases. *Human mutation*, 39(12):1835–1846.

Nishida, K., Hoshide, T., and Fujimura, K. (2012). Improving tweet stream classification by detecting changes in word probability. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 971–980. ACM.

Osborne, T. B. (1894). The proteids of the kidney bean. *Journal of the American Chemical Society*, 16(9):633–643.

Oza, A. M., DiStefano, M. T., Hemphill, S. E., Cushman, B. J., Grant, A. R., Siegert, R. K., Shen, J., Chapin, A., Boczek, N. J., Schimmenti, L. A., et al. (2018). Expert specification of the acmg/amp variant interpretation guidelines for genetic hearing loss. *Human mutation*, 39(11):1593–1613.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Patel, M., DiStefano, M., Oza, A., Hughes, M., Wilcox, E., Hemphill, S., Cushman, B., Grant, A., Siegert, R., Shen, J., et al. (2021). Disease-specific acmg/amp guidelines improve sequence variant interpretation for hearing loss. *Genetics in Medicine*.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peng, J., Xiang, J., Jin, X., Meng, J., Song, N., Chen, L., Abou Tayoun, A., and Peng, Z. (2020). Vip-hl: Semi-automated acmg/amp variant interpretation platform for genetic hearing loss. *Authorea Preprints*.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Piñero, J., Ramírez-Anguita, J. M., Saüch-Pitarch, J., Ronzano, F., Centeno, E., Sanz, F., and Furlong, L. I. (2020). The disgenet knowledge platform for disease genomics: 2019 update. *Nucleic acids research*, 48(D1):D845–D855.

Praagman, J. (1985). Classification and regression trees: Leo breiman, jerome h. friedman, richard a. olshen and charles j. stone the wadsworth statistics/probability series, wadsworth, belmont, 1984, x+ 358 pages.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rehm, H. L., Berg, J. S., Brooks, L. D., Bustamante, C. D., Evans, J. P., Landrum, M. J., Ledbetter, D. H., Maglott, D. R., Martin, C. L., Nussbaum, R. L., et al. (2015). Clingen—the clinical genome resource. *New England Journal of Medicine*, 372(23):2235–2242.

Rentzsch, P., Witten, D., Cooper, G. M., Shendure, J., and Kircher, M. (2019). Cadd: predicting the deleteriousness of variants throughout the human genome. *Nucleic acids research*, 47(D1):D886–D894.

Richards, S., Aziz, N., Bale, S., Bick, D., Das, S., Gastier-Foster, J., Grody, W. W., Hegde, M., Lyon, E., Spector, E., et al. (2015). Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the american college of medical genetics and genomics and the association for molecular pathology. *Genetics in medicine*, 17(5):405–423.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15).

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Salganicoff, M. (1993). Density-adaptive learning and forgetting. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 276–283.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sanger, F. and Coulson, A. R. (1975). A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of molecular biology*, 94(3):441–448.

Scaiewicz, A. and Levitt, M. (2015). The language of the protein universe. *Current Opinion in Genetics & Development*, 35:50–56.

Schlimmer, J. C. and Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1(3):317–354.

Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Schwann, T. (1847). *Microscopical Researches into the Accordance in the Structure and Growth of Animals and Plants*. Ripol Classic.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.

Shearer, A. E., Hildebrand, M. S., and Smith, R. J. (2017). Hereditary hearing loss and deafness overview. *GeneReviews®[Internet]*.

Singh, R. R., Luthra, R., Routbort, M. J., Patel, K. P., and Medeiros, L. J. (2016). Implementation of next generation sequencing in clinical molecular diagnostic laboratories: advantages, challenges and potential. *Expert Review of Precision Medicine and Drug Development*, 1(1):109–120.

Sonnhammer, E. L., Eddy, S. R., Birney, E., Bateman, A., and Durbin, R. (1998). Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Research*, 26(1):320–322.

Spiliopoulou, M., Ntoutsi, E., and Zimmermann, M. (2016). Opinion stream mining. *Encyclopedia of Machine Learning and Data Mining*, pages 1–10.

Tavtigian, S. V., Greenblatt, M. S., Harrison, S. M., Nussbaum, R. L., Prabhu, S. A., Boucher, K. M., and Biesecker, L. G. (2018). Modeling the acmg/amp variant classification guidelines as a bayesian classification framework. *Genetics in Medicine*, 20(9):1054–1060.

Terrapon, N., Weiner, J., Grath, S., Moore, A. D., and Bornberg-Bauer, E. (2013). Rapid similarity search of proteins using alignments of domain arrangements. *Bioinformatics*, 30(2):274–281.

The UniProt Consortium (2017). UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169.

Van Vranken, D. and Weiss, G. A. (2018). *Introduction to bioorganic chemistry and chemical biology*. Garland Science.

Vosoughi, S., Zhou, H., and Roy, D. (2016). Enhanced twitter sentiment classification using contextual information. *arXiv preprint arXiv:1605.05195*.

Wagner, S., Zimmermann, M., Ntoutsi, E., and Spiliopoulou, M. (2015). Ageing-based multinomial naive bayes classifiers over opinionated data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 401–416. Springer.

Wang, K., Li, M., and Hakonarson, H. (2010). Annovar: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164–e164.

Wang, M., Callenberg, K. M., Dalgleish, R., Fedtsov, A., Fox, N. K., Freeman, P. J., Jacobs, K. B., Kaleta, P., McMurry, A. J., Prlić, A., et al. (2018). hgvs: a python package for manipulating sequence variants using hgvs nomenclature: 2018 update. *Human mutation*, 39(12):1803–1813.

Watson, J. D. and Crick, F. H. (1953). Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738.

Xavier, A., Scott, R. J., and Talseth-Palmer, B. A. (2019). Tapes: A tool for assessment and prioritisation in exome studies. *PLoS computational biology*, 15(10):e1007453.

Yang, K. K., Wu, Z., Bedbrook, C. N., and Arnold, F. H. (2018). Learned protein embeddings for machine learning. *Bioinformatics*, 34(15):2642–2648.

Yeo, G. and Burge, C. B. (2004). Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. *Journal of computational biology*, 11(2-3):377–394.

Yu, L., Tanwar, D. K., Penha, E. D. S., Wolf, Y. I., Koonin, E. V., and Basu, M. K. (2019). Grammar of protein domain architectures. *Proceedings of the National Academy of Sciences*, 116(9):3636–3645.