Clustering FunFams using sequence embeddings improves EC purity

Maria Littmann[1,2,*], Nicola Bordin[3], Michael Heinzinger[1,2], Christine Orengo[3,*] & Burkhard Rost[1,4]

1 TUM (Technical University of Munich) Department of Informatics, Bioinformatics & Computational Biology - i12, Boltzmannstr. 3, 85748 Garching/Munich, Germany
2 TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany
3 Institute of Structural and Molecular Biology, University College London, London WC1E 6BT, UK.
4 Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching/Munich, Germany & TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany

* Corresponding author: Maria Littmann: littmann@rostlab.org, http://www.rostlab.org/, Tel: +49-289-17-814 (email rost: assistant@rostlab.org); Christine Orengo: c.orengo@ucl.ac.uk

Abstract

Motivation: Classifying proteins into functional families can improve our understanding of a protein's function and can allow transferring annotations within the same family. Toward this end, functional families need to be "pure", i.e., contain only proteins with identical function. Functional Families (FunFams) cluster proteins within CATH superfamilies into such groups of proteins sharing function, based on differentially conserved residues. 11% of all FunFams (22,830 of 203,639) also contain EC annotations and of those, 7% (1,526 of 22,830) have at least two different EC annotations, i.e., inconsistent functional annotations.
Results: We propose an approach to further cluster FunFams into smaller and functionally more consistent sub-families by encoding their sequences through embeddings. These embeddings originate from deep learned language models (LMs) transferring the knowledge gained from predicting missing amino acids in a sequence (ProtBERT) and have been further optimized to distinguish between proteins belonging to the same or a different CATH superfamily (PB-Tucker). Using distances between sequences in embedding space and DBSCAN to cluster FunFams, as well as identify outlier sequences, resulted in twice as many more pure clusters per FunFam than for a random clustering. 52% of the impure FunFams were split into pure clusters, four times more than for random. While functional consistency was mainly measured using EC annotations, we observed similar results for binding annotations. Thus, we expect an increased purity also for other definitions of function. Our results can help generating FunFams; the resulting clusters with improved functional consistency can be used to infer annotations more reliably. We expect this approach to succeed equally for any other grouping of proteins by their phenotypes.
Availability: The source code and PB-Tucker embeddings are available via GitHub: https://github.com/Rostlab/FunFamsClustering

Key words: Functional families, protein function, CATH, EC numbers, unsupervised learning, contrastive learning, word embeddings, transfer learning

Abbreviations used: DBSCAN, density-based spatial clustering of applications with noise; **d;** dimensions; **EC**, Enzyme Commission; **FunFam**, functional family; **LM**, language model; **NLP**, natural language processing

Introduction

Knowledge about the function of a protein is crucial for a wide array of biomedical applications and the classification of protein sequences into functional families can help transferring annotations from a functional family to an uncharacterized protein. Functional families can also reveal insights into the evolution of function through sequence changes [1]. To gain meaningful insights into protein functionality through functional families, it is important that those families are consistent, i.e., only contain functionally similar proteins.

CATH FunFams [2, 3] provide a functional sub-classification of CATH superfamilies [4, 5]. Superfamilies are the last level (H) in the CATH hierarchy; they group sequences which are related by evolution, often referred to as homologos. However, proteins in one superfamily can still be functionally and structurally diverse. Functional families (FunFams) provide a further sub-classification of superfamilies into coherent subsets of proteins with the same function. FunFams can be used to predict function on a per-protein level as described through Gene Ontology (GO) terms [6, 7], to predict functional sites [8], or to improve binding residue predictions by combining predictions of one FunFam in a consensus prediction [9].

The Enzyme Commission number (EC number) [10] numerically classifies enzymatic functions based on the reactions they catalyze. It consists of four levels and each level provides a more specific description of function than the previous one. The function of two proteins is more similar, the more levels of their two EC numbers are identical, particularly, for the levels EC3 and EC4 which describe the chemical reaction and its substrate specificity.

For 22,830 FunFams (11% of all), annotations for EC numbers for all four levels are available at least for one member. By design, proteins from the same FunFam should share the same EC class (annotated up to level 4). However, 1,526 FunFams (7% of 22,830) accounting for 16% of all sequences in the 22,830 FunFams with EC annotations have more than one annotation, and 180 (1% of 22,830) accounting for 2% of the sequences have even four or more different annotations (Fig. S1 in Supporting Online Material (SOM)). Different EC annotations within one FunFam could originate from moonlighting enzymes, i.e., enzymes with multiple functions [11]. Assuming the moonlighting enzyme to have two EC numbers, only one would be inconsistent with the other FunFam members rendering that FunFam inconsistent. However, different EC annotations can also result from impurity, i.e., FunFams containing proteins with different functions. Splitting FunFams further could provide a more fine-grained and consistent set of functionally related proteins.

Over the last few years, novel representations (embeddings) for proteins have emerged from adapting language models (LMs) developed for natural language processing (NLP) to protein sequences [12-16]. These embeddings are learnt solely from protein sequences without any additional annotations (self-supervised) using either auto-regressive pre-training (predicting the next amino acid, given all previous amino acids in a sequence, e.g., ELMo [17] or GPT [18]) or masked language modeling (reconstructing corrupted amino acids from the sequence, e.g., BERT [19]). To do well in those tasks, the LM is forced to learn frequently co-occurring sequence patches as well as more complex protein features such as those underlying secondary structure formation [20] as it is not possible to learn all possible amino acid permutations over the large set of protein sequences used for training. Features learnt implicitly by these models can be transferred to any task requiring protein representations by extracting the hidden states of the LM for a given protein sequence (transfer learning). It was shown previously that those learnt representations – referred to as embeddings – capture

higher-level features of proteins, including aspects of protein function beyond what is available through traditional comparisons using sequence similarity or homology-based inference [21]. Therefore, we hypothesized that this orthogonal perspective – using embedding rather than sequence space to transfer annotations – might help to find functionally consistent sub-groups within protein families built using sequence similarity.

Here, we proposed a clustering approach to identify clusters in FunFams that are more consistent in terms of shared functionality. To this end, shared functionality was defined as sharing the same EC annotation up to the fourth level (i.e., completely identical EC numbers). We represented protein sequences as embeddings, i.e., fixed-size vectors derived from pre-trained LMs. We used the LM ProtBERT [15] to retrieve the initial embeddings, and applied contrastive learning to map them onto a new embedding space where proteins within one CATH superfamily were closer together than proteins from different superfamilies. The resulting embeddings are called PB-Tucker. Clustering was then performed based on the Euclidean distances between those embeddings using DBSCAN [22]. Within each FunFam DBSCAN identified clusters as dense regions in which all sequences were close to each other in embedding space; it classified proteins as outliers if they were not close to other sequences in the FunFam. That allowed the identification of (i) a more fine-grained clustering of the FunFams, and (ii) single sequences which might have been falsely assigned to this FunFam. Analyzing whether or not embedding-based clustering reduced the number of different EC annotations in a FunFam allowed validating our new approach.

Methods

FunFams dataset. The current version of CATH (v4.3) holds 4,328 superfamilies split into 212,872 FunFams. The FunFams generation process, albeit changing through time, consists of various steps, starting with the clustering of all sequences within a CATH superfamily at 90% sequence similarity, encoding these clusters in Hidden Markov Models and creating a relationship tree between all clusters using GeMMA [23] and HHsuite [24]. Subsequently, CATH-FunFHMMer [7] is applied to transverse the tree and GroupSim [25] conservation patterns are employed to merge or cut the tree branches to obtain the largest possible alignment that is functionally pure. CATH FunFams have higher functional purity than CATH superfamilies and conserved residues are enriched in functional sites [7].

EC annotations and EC purity. EC annotations for the FunFams dataset were obtained using the UniProt [26] SPARQL API and cross-assigned to all UniProt IDs available within the FunFams. Since proteins in the same FunFam are assumed to share a function, we expect all proteins in one FunFam to have the same EC number(s). If not, the FunFam is considered *impure,* i.e., it contains sequences which do not belong to this functional family. Impurity can naively be defined as any FunFam with more than one EC number. However, some proteins are annotated with multiple EC numbers. These proteins might actually execute multiple functions (moonlighting) [11] or annotations might be wrong. Such an impurity is not caused by an error in the creation of the FunFams and cannot be removed by further clustering the FunFams. Instead, the naïve definition of impurity considers FunFams with one protein with two different EC numbers as impure even if all other proteins share the same two EC numbers, i.e., the annotations would be consistent and therefore, the FunFam should be considered as

pure. Consequently, we refined the definition considering FunFams as impure if one or more relatives were annotated to additional EC numbers different to the other family members. We only considered EC annotations with all four levels; all others were treated like those without annotation.

Protein representation. We used ProtBERT [15] to create fixed-length vector representations, i.e., vectors with the same number of dimensions irrespective of protein length. ProtBERT uses the architecture of the LM BERT [19] which applies a stack of self-attention [27] layers for masked language modeling (Supporting Online Material Section 1 (SOM_1) for details). Fixed-length vectors were derived by averaging over the representations of each amino acid extracted from its last layer. This simple global average pooling provides an effective baseline [12, 15, 20]. In the following, *ProtBERT* refers to this representation.

In order to capture the CATH hierarchy more explicitly, ProtBERT representations were mapped to a new embedding space via contrastive learning. While supervised learning requires phrasing a prediction task as e.g., a classification or regression task, contrastive learning only requires some notion of similarity between samples. This similarity is used to learn a new vector space that clusters similar samples while dissimilar items are separated. Similarity can be defined between sample triplets potentially capturing their triangular relation; in this case an *anchor* sample is given together with a *positive* and *negative* sample with the positive being more similar to the anchor than the negative. The network then learns to push anchor and positive toward each other while pushing anchor and negative apart. While mapping a CATH-like hierarchy onto supervised classification is challenging, using a hierarchy to define relative similarity between triplets is straightforward as anchor and positive only need to share one level more in the hierarchy than anchor and negative. Toward this end, ProtBERT representations were projected in two steps from 1024-dimensions (1024-d) to 128-d using CATH v4.3 [3] for training the two-layer neural network (details in SOM_1). In the following, we call these new 128-d embeddings *PB-Tucker* (Heinzinger et al., unpublished). PB-Tucker has been trained to differentiate CATH superfamilies and seemed to better capture functional relationships between proteins in one superfamily than the original ProtBERT (data not shown; SOM_1 for more details).

Clustering. Representing sequences as PB-Tucker embeddings, we calculated the Euclidean distance between all sequences within one FunFam. The distance *d* between two embeddings *x* and *y* was defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{1024}(x_i - y_i)^2} \quad \text{(Eqn. 1)}$$

Based on these distances, we clustered all sequences within one FunFam using the implementation of DBSCAN [22] in scikit-learn [28] For a set of data points, DBSCAN identifies dense regions, i.e., regions of points that are close to each other, and classifies these regions as clusters. Data points not close to enough other data points are classified as outliers. DBSCAN is based on the identification of *core points* that seed a cluster; all points within a certain distance of the core point are added to this cluster. Two free parameters were optimized: (1) The number of neighbors *n* (including the point itself) a point needs to have to

become “core point”; *n* implicitly controls the size and number of clusters, and (2) the distance cutoff θ. Data points A and B are considered close, if d(A,B)< θ. For our application, DBSCAN has two major advantages: (1) The number of clusters does not have to be set *a priori*, and (2) clustering and outlier detection are simultaneous.

If not stated otherwise, we used the default *n*=5 although it has been suggested to use values between *n*=D+1 and *n*=2*D-1 where D is the number of dimensions [29] With d=128 for the PB-Tucker embeddings that implies *n*=255. Since FunFams vary in size, *n* might be adjusted to that size. For five superfamilies, we tested, in addition to *n*=5, *n*=129, *n*=255 as fixed neighborhood sizes, as well as *n*=0.05*|F|, *n*=0.1*|F|, *n*=0.2*|F| (|F|=number of sequences in FunFam) as variable neighborhood sizes dependent of the size of the FunFam.

Observing differences in the distances between the members of different superfamilies (Fig. S2), it appeared best to choose superfamily-specific values for θ. Initially, we wanted to determine a distance threshold reflecting the expected distance between any two members of the same FunFam. However, large distances between members in one FunFam might reveal impurity rather than a generic width of a family. Instead, we computed the median over those distances for all FunFams in one superfamily and used this value for each FunFam. This way, the value still reflects the expected distance between two members of a FunFam, but the effect of large distances due to impurity should be averaged out by considering all FunFams in a superfamily. In detail, for each member in each FunFam in a superfamily, we calculated its average distance to all other members of that FunFam (distance distribution for five superfamilies in Fig. S2). Given the distribution of these average sequence distances, we chose the median distance as θ, i.e., we chose a distance cutoff so that 50% of all sequences in a superfamily were on average within a distance of θ to all other sequences in the same FunFam. Decreasing θ raises outliers and yields smaller clusters while increasing θ reduces outliers and yields larger clusters.

Measuring purity of clustered FunFams. To estimate whether the clustering of an impure FunFam led to more consistent sub-families, we calculated the percentage of pure clusters. Clusters with no EC annotation were excluded. For each FunFam, we calculated the clusters with one single EC annotation as percentage of all clusters with EC annotations and defined this measure as the purity of a FunFam (Eqn. 2). We then defined the percentage of completely pure FunFams as the percentage of FunFams with a purity of 100.

$$Purity(F) = \frac{\#clusters_{pure}}{\#clusters_{with\ ECs}} \cdot 100 \qquad \text{(Eqn. 2)}$$

We also calculated the purity of a FunFam in terms of its size, i.e., the number of sequences contained in it:

$$Purity_{Sequences}(F) = \frac{\#sequences\ in\ pure\ clusters}{\#\ sequences\ in\ clusters\ with\ ECs} \cdot 100 \qquad \text{(Eqn. 3)}$$

Confidence intervals (CIs). 95% symmetric confidence intervals (CIs) were calculated from 1, 000 bootstrap samples with replacement to indicate the spread of data and certainty of average values.

Final dataset. To construct the dataset used in this analysis, we extracted all superfamilies with at least one impure FunFam, i.e., at least one FunFam with more than one EC annotation. Since embeddings could only be computed for continuous sequences, we excluded sequences with multiple segments. After this removal, some FunFams became orphans (single member) and were also excluded. This led to a final dataset of 458 superfamilies (10.6% of all superfamilies) with 110,876 FunFams (52.1%) and 13,011 (6.1%) with EC annotations. Those 13,011 FunFams accounted for 20% of all proteins in the FunFams (1,669,245 sequences). All FunFams in a superfamily were used to determine a reasonable distance cutoff for clustering while clustering was only performed for FunFams with EC annotations. FunFams without EC annotations could have been clustered, too. However, since EC annotations served as criterion for evaluation, only FunFams with such annotations were clustered to save computational time and hence energy.

Results & Discussion

Embedding clusters increased EC purity. We began with 13,011 FunFams (6% of all) with at least one EC annotation. Of these, 1,273 (10%) contained more than one EC annotation (impure FunFams). Applying DBSCAN to all EC annotated FunFams, we split these into 26,464 clusters (21,546 for pure and 4,918 for impure FunFams). On average, 4.5% (95% confidence interval (CI): [4.4%; 4.6%]) of the sequences in a FunFam were classified as outliers (Table S1 in Supplementary Online Material (SOM)). 63% of the DBSCAN clusters contained proteins with EC annotations; only 4% of those contained more than one EC annotation (compared to 10% in all FunFams; Fig. 1A). Only 10% of all proteins (155,044 of 1,593,567) belonged to clusters with more than one EC annotation compared to 21% (356,565 of 1,668,273) for FunFams (Fig. 1B). Consequently, a larger fraction of clusters was pure (i.e., contained one EC annotation) than of FunFams both in terms of numbers of clusters and numbers of proteins (Fig. 1).

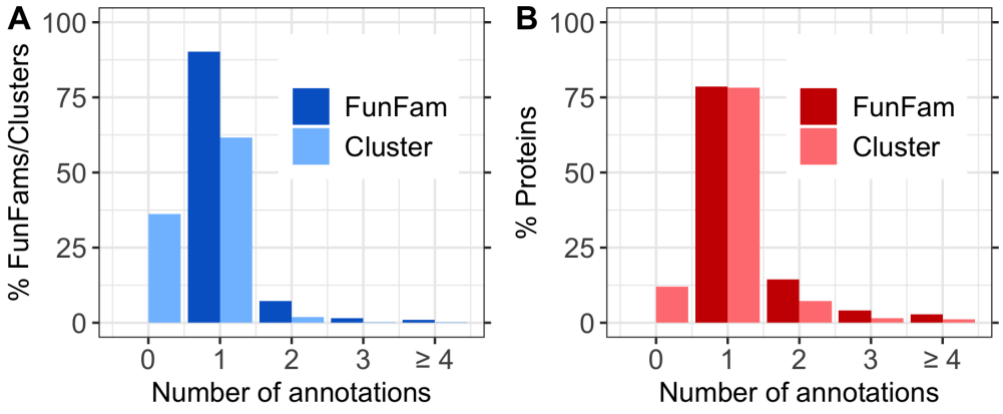

Fig. 1: EC purity for FunFams and embedding clusters. This analysis considered 13,011 FunFams with EC annotations. Panel **A** shows the distribution of all families (FunFam/Clusters), i.e., the percentage of FunFams and embedding-based clusters with *n* EC annotations (n≥1 for FunFams and n≥0 for new clusters; note: bars left and right of integer values *n*, not separated by a white space denote *n* annotations). Panel **B** shows the distribution of all proteins, i.e., the percentage of proteins in families

(FunFam/Cluster) with *n* EC annotations. This number does not reveal how many proteins have an EC annotation. Of the 13,011 FunFams, 10% were impure, i.e., they contained more than one EC annotation (100-value for dark blue bar at 1 in panel A), and 21% of all proteins were part of these impure FunFams. After embedding-based split of FunFams, 64% (16,906) of the resulting clusters contain ECs (100-light blue bar at 0) and 4% (606) of those 16,906 were annotated to more than one EC accounting for 11% of proteins in clusters with ECs.

To further understand the extent to which the clustered FunFams provide a functionally more consistent subset, we determined for each impure FunFam, the fraction of clusters that were pure (Methods). To begin: 37% of all clusters had no EC annotated proteins and were excluded from further analysis. Of the remaining 16,906 clusters (63%), 22% were impure, i.e., contained more than one EC annotation. On average, 63% (CI: [60%; 66%]) of the clusters for a FunFam were pure (Fig. 2; dashed blue line) accounting for 58% (CI: [55%; 61%]) of all proteins (Fig. 2; dotted red line). 52% of all impure FunFams were split into completely pure clusters, i.e., for every other FunFam, the embedding-split clustered into functionally consistent sub-families (Fig. 2, right most blue point “100% Pure Clusters”) accounting for 38% of all proteins (Fig. 2, right most red point). This measure gave conservative estimates as it only considered completely pure clusters, ignoring improvements through reduction of EC annotations, e.g., when a group had originally *m+1* annotations and the clustering improved to *m*, this improvement was ignored for all *m>1*.

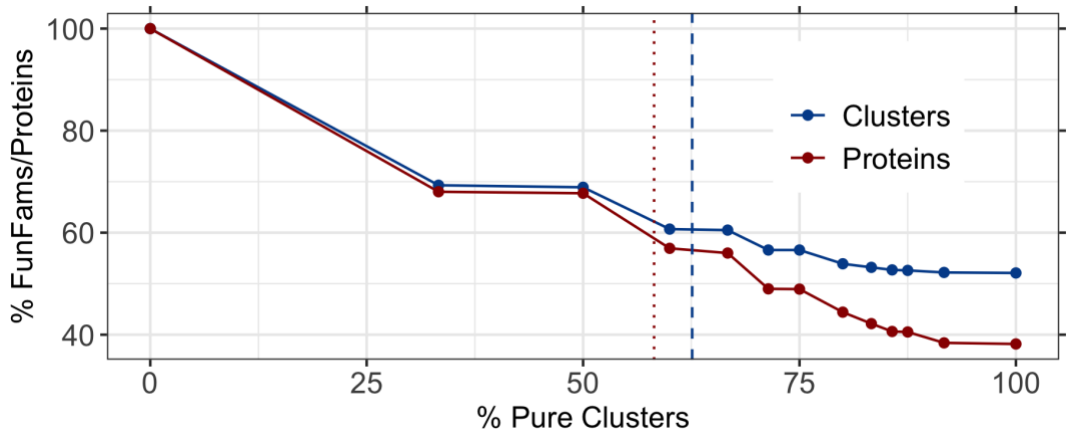

Fig. 2: Fraction of pure clusters for impure FunFams. We show the percentages of all FunFams (blue line) or of all proteins (red line) in FunFams at levels of increasing cluster purity (Eqns. 2, 3). On average, 63% of clusters for a FunFam were pure (dashed blue line) accounting for 58% of the proteins (dotted red line). 52% of impure FunFams were split only into pure clusters (right most blue point) accounting for 38% of the proteins.

Improving EC purity without over-splitting. While splitting impure FunFams through embedding-based clustering based clearly improved the EC purity of these FunFams, we wanted to avoid over-splitting. Trivially, the more and smaller clusters a FunFam is split into, the more likely it is that these clusters are pure. In the non-sense extreme case of having N clusters for N sequences (each sequence a cluster), all clusters are trivially pure. One constraint to avoid generating too many clusters (over-splitting) is to do substantially better than by randomly splitting into the same number of clusters. We computed the random

clustering using the same cluster sizes and outlier numbers as realized by the embedding-based clustering. Embedding-based clustering outperformed random (Fig. 3): More than twice as many clusters were impure for random than for embedding-based clustering (Fig. 3A, 47±1% vs 22±1%); the average purity of a FunFam was almost two times higher for embedding-based clustering than for random (Fig. 3B, 63±3% vs. 38±5%), and 3.5 times more FunFams were split into exclusively pure clusters by the embedding-based clustering (Fig. 3C, 52±3% vs 15±1%). This corresponded to 4.8% (CI: [4.4%; 5.2%]) of all proteins clustered into pure clusters at random compared to 38% (CI: [33%; 43%]) of all proteins for embedding-based clustering, i.e., an over 7-fold increase (Fig. 3C, red bars).

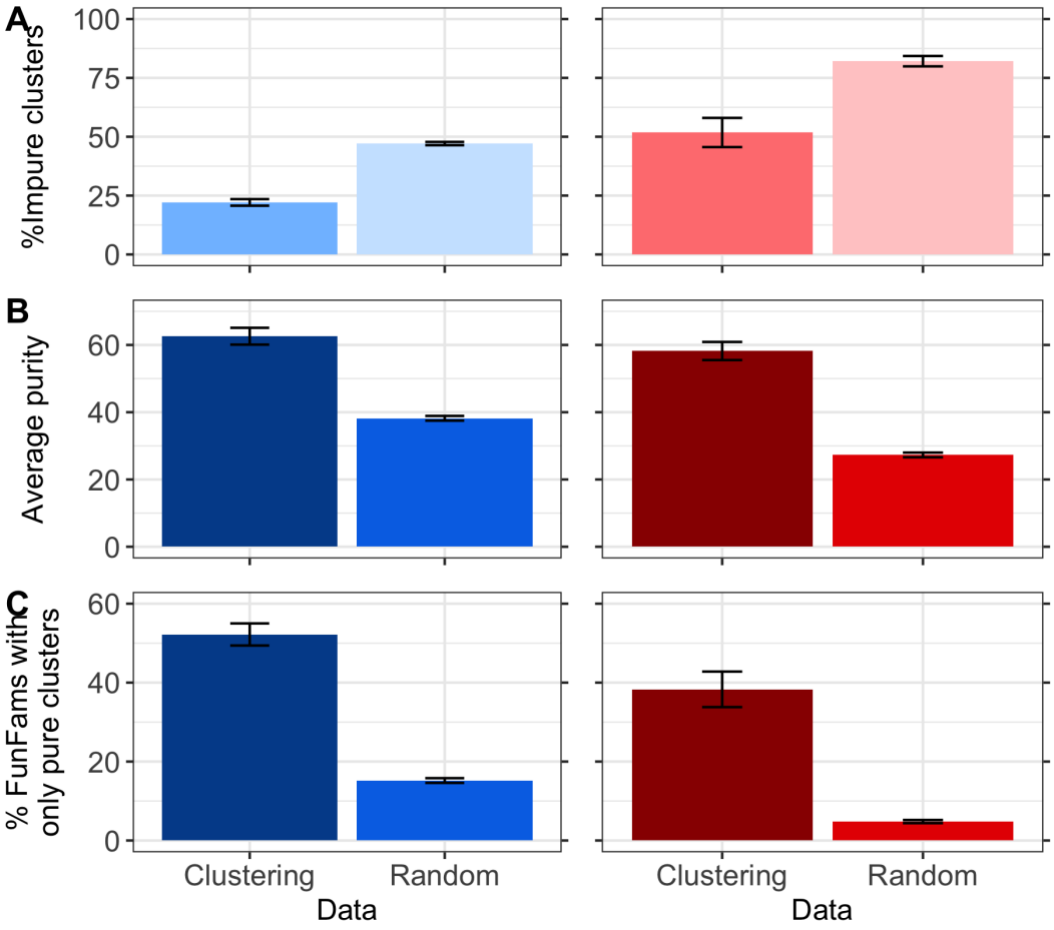

Fig. 3: Embedding-based clusters improve EC purity over random. Random clusters were computed using the same cluster size and outlier number realized by the embedding-based clustering, but the FunFam members were randomly assigned to one of these clusters or were classified as outliers. Plots on the left (blue colors) show percentages of FunFams/Clusters, plots on the right show percentages of proteins. **A.** The fraction of impure clusters was higher for the random clustering than for our clustering (29% vs 12%). **B.** Through DBSCAN embedding-based clustering, each impure FunFam was, on average, split into 63% pure clusters while for the random clustering, the average purity was only 38%. **C.** More than half of all FunFams (53%) were split only into pure clusters for embedding-based clustering but only 15% for a random clustering. Error bars indicate symmetric 95% confidence intervals.

An ideal split of impure FunFams generates clusters defined by single EC numbers, i.e., all cluster members share the same EC annotation and all proteins with the same EC annotation end up in the same cluster. Ignoring the latter leads to over-splitting. For the embedding-based clustering, 81% of the ECs occurred in one cluster (Fig. 4). However, some of the outliers had EC annotations. When also counting those (as single member clusters), the percentage of EC-exclusive clusters dropped to 63% (Fig. 4). These results suggested the embedding-based clustering to have largely avoided over-splitting. Nevertheless, 8% of all experimentally known EC numbers were annotated to proteins from at least three different clusters (17% if including outliers; Fig. 4) and some (10%) of the outliers shared the EC number with the cluster from which they had been removed. This might indicate over-splitting or suggest a more fine-grained functional distinction between those proteins than is captured in the fourth EC level.

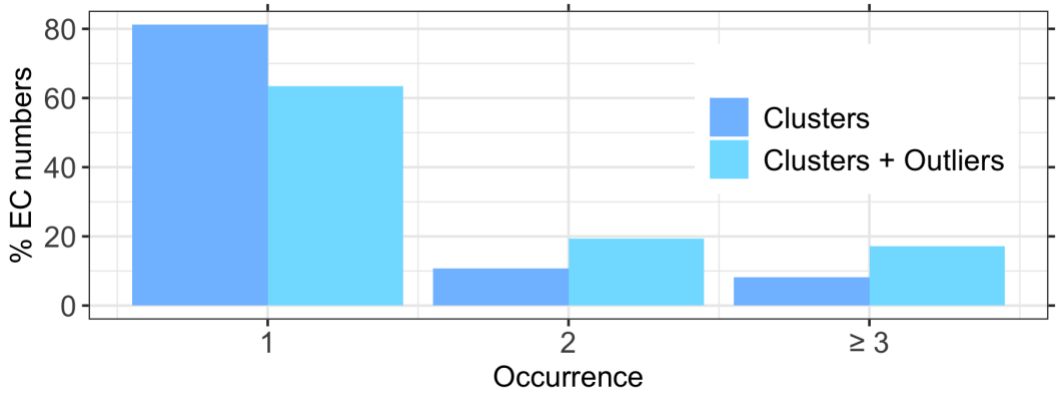

Fig. 4: Most EC numbers only occur in one cluster. For each EC number in a FunFam, we counted the number of embedding-based clusters in which it occurred to gauge potential over-splitting. 81% of the ECs only occurred in one cluster (darker bars). If we considered outliers as clusters with one member, this number dropped to 63%. These results suggest that the clustering did not over-split the FunFams and that functionally related proteins ended up in the same cluster.

If the increased purity through clustering had been a random effect, the embedding space clustering would be EC-independent. If so, we expect no difference in the distributions of embedding distances between pure and impure FunFams, and a similar number of clusters and outliers. However, pure FunFams were, on average, split into only two clusters, while impure FunFams were split into four clusters (Table S1). The number of clusters can, thus, indicate whether a FunFam is impure or not, i.e., if a FunFam is split into many clusters, it should be considered for further manual inspection to establish whether all proteins were correctly assigned to this functional family (more details in SOM_2.1).

Different levels of EC annotations gave similar results. Up to this point, we only distinguished whether two proteins were annotated to the same or to different EC numbers, ignoring that two proteins with ECs A.B.C.X and A.B.C.Y more likely have similar molecular function than a pair with A.* and D.*. Pairs of the first type (difference only in 4th-EC level) will, on average, be more sequence similar than pairs of the second type (different EC numbers at the top level). Most impure FunFams were impure due to differences on the fourth level of EC annotations (Fig. S4). Although we analyzed the clustering at higher levels of the EC

classification, the results were inconclusive, probably due to data sparsity (SOM_2.2 for more details).

Details of parameter choice mattered. For a more detailed analysis of particular details of our method, in particular, for the choice of embeddings and clustering parameters, we chose five superfamilies with diverse properties (CATH identifiers: 3.40.50.150, 3.20.20.70, 3.40.47.10, 3.50.50.60, 1.10.630.10; SOM_3).

More consistent clustering from PB-Tucker. When using ProtBERT embeddings to cluster the five chosen superfamilies, the number of clusters and outliers was smaller, but the fraction of impure clusters was higher than when using the default PB-Tucker embeddings (19% for ProtBERT vs 13% for "default"; Table S4). The average purity was also higher for PB-Tucker ("default" = 59%) than for ProtBERT (51%) (Table S4). Thus, PB-Tucker appeared superior in capturing functional differences, yielding a more fine-grained and pure clustering.

Smaller distance thresholds led to smaller and purer clusters. The distance threshold q of DBSCAN defines whether or not two points are close enough to each other to be grouped. For the default clustering, we chose the median distance between all proteins for each superfamily (Methods). The observed distribution of distances (Fig. S2) suggested choosing superfamily-specific thresholds. As expected, the smaller q, the more clusters and outliers will result ("q =1st quartile" vs "default"; Table S4). Largely due to splitting FunFams into more clusters at smaller q, the resulting clusters were seemingly purer with only 4% impure clusters (vs. 13%) and an average purity of 83% (vs. 59%) (Table S4). In contrast, larger q thresholds (here the 3rd quartile) affected fewer, more impure clusters (Table S4). Thus, the choice of q highly influences the clustering results. For some applications, lower values of q might be best to obtain a large, highly consistent set of small sub-families that can serve e.g., as seed to further extend those sub-families to larger functionally related families. Also, especially for FunFams for which using a larger cutoff did not result in any clusters and only a small number of outliers, decreasing the distance threshold can help to still identify which sequences might cause impurity.

Default neighborhood size resulted in best clustering. DBSCAN forms clusters around "core points" which are points with at least *n* neighbors. For the five superfamilies, we tested fixed neighborhood sizes of *n*Î[5;129;255] and variable neighborhood sizes dependent on the size of the FunFam *n*=x*|F|, with |F| as the number of proteins in a FunFam and xÎ[0.01;0.1;0.2] (Methods). While *n*=129 and *n*=255 were in the range of what is recommended for *n* [29], the clustering was worse than for the default parameter (*n*=5) (Table S4). Specifically, the number of outliers exploded for these large neighborhood sizes (Table S4, Fig. S5).

Since FunFams differ substantially in the number of proteins, we hypothesized that – similar as for q – it could be reasonable to choose a different *n* for each FunFam. However, this did not improve compared to the default clustering (Table S4); the default *n*=5 was a good choice.

No consistent influence of level of EC annotation. Assessing how well our clustering approach worked depending on the level on which EC annotations were different (E.g., for a given level, for example EC level 3, we checked that annotations for level 1 and 2 were consistent) did not reveal a consistent trend (Fig. S4). The results for the five chosen superfamilies were similar (Fig. S6) underlining the more general findings that the level of EC annotation causing impurity

did not tremendously influence the results of the clustering (see SOM for a more detailed analysis). The performance is rather impacted by other factors like the presence of moonlighting proteins or missing annotations. We applied a rather conservative definition of purity: If one protein is annotated to two EC numbers and another protein in the same cluster is only annotated to one of those two, we considered this cluster impure. We would argue that it makes sense to group all proteins with two annotations in one cluster and proteins with only one annotation in another. However, those proteins also clearly share some function and considering those FunFams or clusters as impure is probably too strict. Also, we cannot be sure whether proteins with only one annotation are correctly annotated or are missing an annotation. In general, missing annotations limited our approach. Many resulting clusters did not contain any sequence with an EC annotation making it hard to assess whether those clusters were pure or not. Assessing how well our clustering approach worked depending on the level on which EC numbers differed in impure FunFams (e.g., for EC level 3, annotations for levels 1 and 2 were consistent) did not reveal a consistent trend (Fig. S4). The results for the five chosen superfamilies were similar (Fig. S6) underlining the more general findings that the level of EC annotation causing impurity did not crucially affect the embedding-based clustering (SOM_3.2). Instead, the performance was likely impacted more by other factors such as the presence of moonlighting proteins or missing annotations. We applied a rather conservative definition of purity: If one protein is annotated to two EC numbers and another protein in the same cluster is only annotated to one of those two, we considered this cluster impure. We would argue that it makes sense to group all proteins with two annotations in one cluster and proteins with only one annotation in another. However, those proteins also clearly share some function and considering those FunFams or clusters as impure is probably too strict. Also, we cannot be sure whether proteins with only one annotation are correctly annotated or are missing an annotation. In general, missing annotations limited our approach. Many resulting clusters did not contain any protein with an experimental EC annotation making it hard to assess whether those clusters were pure or not.

Clustering increased purity of ligand binding. Another way to assess the purity of molecular protein function within a group of proteins is by comparing the extent to which they are similar in terms of ligand-binding. We extracted bound ligands from BioLip [30] and only considered annotations defined as the cognate ligand [31] (SOM_1.2). Of the 13,011 FunFams considered so far, 950 (7%) contained any annotation about a ligand bound, and of those 950, 158 (17%) were annotated with more than one different ligand. Embedding-based clustering split 33% of these FunFams into clusters with only one type of ligand, i.e., “pure” clusters (compared to 52% for EC level 4) and an average purity of 36% (compared to 63% for ECs). Although ligand annotations remained limited, these results confirmed that embedding-based clustering increased functional purity of FunFams for an aspect of function not used during method development.

Conclusions

FunFams [4, 5] provide a high-quality sub-classification of CATH superfamilies into families of functionally related proteins [3, 32]. However, some FunFams are impure and 7% of all FunFams with EC annotations contain at least two different ECs (Fig. S1). Here, we introdued a novel approach toward clustering proteins through embeddings derived from the LM ProtBERT [15] and further optimized to capture relationships between proteins within one CATH superfamily (called *PB-Tucker*). Similarity between embeddings can capture information different from what is captured by sequence similarity. In particular, it can reveal new functional relations between proteins [21]. Clustering all FunFams with more than one EC annotation (impure FunFams) using DBSCAN [22] reduced the percentage of impure clusters to 22% (95% confidence interval (CI): [21%, 23%]). An impure FunFam was on average split into 63% pure clusters (CI: [60%: 66%]) and more than half (53%, CI: [50%; 56%]) of all impure FunFams were split into fully pure sub-families (Fig. 2). This corresponded to a four-fold increase over random clustering (Fig. 3B). In terms of number of proteins (rather than number of clusters), the increase was almost ten-fold. Only 4.8% (CI: [4.4%; 5.2%]) of the proteins were in FunFams split into pure clusters for random while this number rose to 38% (CI: [33%; 43%]) for the *PB-Tucker* embedding-based clustering.

A more detailed analysis of five hand-picked superfamilies (Table S2) showed that the default choices for the DBSCAN parameters were reasonable (Figs. S4 & S5, Table S4), with the default n=5 to define the number of neighbors for a point to be considered a core point and the distance threshold q determined automatically based on the median distance between proteins within one FunFam.

Restricting the analysis to experimental EC annotations limited the validation of our approach to a small fraction (6.1%) of all FunFams and even for those FunFams, most EC annotations remain unknown. Nevertheless, we have shown that our approach could capture more fine-grained functional relationships and enabled splitting FunFams into more functionally consistent sub-families. Especially for FunFams without many known functional annotations, our clustering can be used to (i) investigate whether or not the family could be impure based on the number of clusters resulting from the embedding-based split, or (ii) more safely infer functional annotations between members of one functional cluster than between members of one FunFam. We presented evidence suggesting that the findings for EC annotations will hold for other aspects of protein function, e.g., for binding. While we only applied this approach to FunFams using embeddings optimized for CATH, this clustering could be applied to any database of functional families using a more generalized version of those embeddings.

Acknowledgements

Thanks to Tim Karl and Inga Weise (both TUM) for invaluable help with technical and administrative aspects of this work. We would like to acknowledge Ian Sillitoe (UCL) for helpful comments on EC data. Last, but not least, thanks to all maintainers of public databases and to all experimentalists who enabled this analysis by making their data publicly available.

This work was supported by the Bavarian Ministry of Education through funding to the TUM, by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung), and by two grants from BMBF (031L0168 and Program "Software Campus 2.0 (TUM): 01IS17049) as well as by a grant from Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Titan GPU used for this research. Nicola Bordin acknowledges financial support from the Biotechnology and Biological Sciences Research Council (UK) [BB/R009597/1].

References

[1] S. S. Hannenhalli and R. B. Russell, "Analysis and prediction of functional sub-types from protein sequence alignments," *J Mol Biol*, vol. 303, no. 1, pp. 61–76, Oct. 2000, doi: 10.1006/jmbi.2000.4036.

[2] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "CATH--a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–108, Aug. 1997, doi: 10.1016/s0969-2126(97)00260-8.

[3] "CATH: Protein Structure Classification Database at UCL." https://www.cathdb.info/ (accessed Nov. 02, 2020).

[4] I. Sillitoe *et al.*, "CATH: increased structural coverage of functional space," *Nucleic Acids Res.*, no. gkaa1079, Nov. 2020, doi: 10.1093/nar/gkaa1079.

[5] S. Das, D. Lee, I. Sillitoe, N. L. Dawson, J. G. Lees, and C. A. Orengo, "Functional classification of CATH superfamilies: a domain-based approach for protein function annotation," *Bioinformatics*, vol. 32, no. 18, p. 2889, Sep. 2016, doi: 10.1093/bioinformatics/btw473.

[6] I. Sillitoe *et al.*, "New functional families (FunFams) in CATH to improve the mapping of conserved functional sites to 3D structures," *Nucleic Acids Res*, vol. 41, no. Database issue, pp. D490-8, Jan. 2013, doi: 10.1093/nar/gks1211.

[7] N. Zhou *et al.*, "The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens," *Genome Biol*, vol. 20, no. 1, p. 244, Nov. 2019, doi: 10.1186/s13059-019-1835-8.

[8] S. Das *et al.*, "CATH FunFHMMer web server: protein functional annotations using functional family assignments," *Nucleic Acids Res*, vol. 43, no. W1, pp. W148-53, Jul. 2015, doi: 10.1093/nar/gkv488.

[9] S. Das, H. M. Scholes, and C. A. Orengo, "CATH functional families predict protein functional sites," 2020, doi: 10.1101/2020.03.23.003012.

[10] L. Scheibenreif, M. Littmann, C. Orengo, and B. Rost, "FunFam protein families improve residue level molecular function prediction," *BMC Bioinformatics*, vol. 20, no. 1, p. 400, Jul. 2019, doi: 10.1186/s12859-019-2988-x.

[11] E. C. Webb, *Enzyme nomenclature 1992: recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*. Academic Press, 1992.

[12] C. J. Jeffery, "Moonlighting proteins: old proteins learning new tricks," *Trends Genet*, vol. 19, no. 8, pp. 415–7, Aug. 2003, doi: 10.1016/S0168-9525(03)00167-7.

[13] M. Heinzinger *et al.*, "Modeling aspects of the language of life through transfer-learning protein sequences," *BMC Bioinformatics*, vol. 20, no. 1, p. 723, Dec. 2019, doi: 10.1186/s12859-019-3220-8.

[14] R. Rao *et al.*, "Evaluating Protein Transfer Learning with TAPE," *ArXiv190608230 Cs Q-Bio Stat*, Jun. 2019, Accessed: Nov. 03, 2020. [Online]. Available: http://arxiv.org/abs/1906.08230.

[15] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, "Unified rational protein engineering with sequence-based deep representation learning," *Nat. Methods*, vol. 16, no. 12, Art. no. 12, Dec. 2019, doi: 10.1038/s41592-019-0598-1.

[16] A. Elnaggar *et al.*, "ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing," *bioRxiv*, p. 2020.07.12.199554, Jul. 2020, doi: 10.1101/2020.07.12.199554.

[17] A. Madani *et al.*, "ProGen: Language Modeling for Protein Generation," *bioRxiv*, p. 2020.03.07.982272, Mar. 2020, doi: 10.1101/2020.03.07.982272.

[18] M. E. Peters *et al.*, "Deep contextualized word representations," *ArXiv180205365 Cs*, Mar. 2018, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1802.05365.

[19] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," p. 12.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *ArXiv181004805 Cs*, May 2019, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1810.04805.

[21] A. Rives *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *bioRxiv*, p. 622803, Jan. 2020, doi: 10.1101/622803.

[22] M. Littmann, M. Heinzinger, C. Dallago, T. Olenyi, and B. Rost, "Embeddings from deep learning transfer GO annotations beyond homology," *bioRxiv*, p. 2020.09.04.282814, Oct. 2020, doi: 10.1101/2020.09.04.282814.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996, vol. 96, pp. 226–231.

[24] D. A. Lee, R. Rentzsch, and C. Orengo, "GeMMA: functional subfamily classification within superfamilies of predicted protein structural domains," *Nucleic Acids Res.*, vol. 38, no. 3, pp. 720–737, Jan. 2010, doi: 10.1093/nar/gkp1049.

[25] M. Steinegger, M. Meier, M. Mirdita, H. Vöhringer, S. J. Haunsberger, and J. Söding, "HH-suite3 for fast remote homology detection and deep protein annotation," *BMC Bioinformatics*, vol. 20, no. 1, p. 473, Sep. 2019, doi: 10.1186/s12859-019-3019-7.

[26] J. A. Capra and M. Singh, "Characterization and prediction of residues determining protein functional specificity," *Bioinformatics*, vol. 24, no. 13, pp. 1473–1480, Jul. 2008, doi: 10.1093/bioinformatics/btn214.

[27] T. U. Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D506–D515, Jan. 2019, doi: 10.1093/nar/gky1049.

[28] "UniProt." https://sparql.uniprot.org/ (accessed Nov. 12, 2020).

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *ArXiv14090473 Cs Stat*, May 2016, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1409.0473.

[30] D. Shen *et al.*, "Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms," *ArXiv180509843 Cs*, May 2018, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1805.09843.

[31] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[32] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, Jun. 1998, doi: 10.1023/A:1009745219419.

[33] J. Yang, A. Roy, and Y. Zhang, "BioLiP: a semi-manually curated database for biologically relevant ligand–protein interactions," *Nucleic Acids Res.*, vol. 41, no. Database issue, pp. D1096–D1103, Jan. 2013, doi: 10.1093/nar/gks966.

[34] J. D. Tyzack, L. Fernando, A. J. M. Ribeiro, N. Borkakoti, and J. M. Thornton, "Ranking Enzyme Structures in the PDB by Bound Ligand Similarity to Biological Substrates," *Structure*, vol. 26, no. 4, pp. 565-571.e3, Apr. 2018, doi: 10.1016/j.str.2018.02.009.