Deliverable D7.8

Project Title:	World-wide E-infrastructure for structural biology	
Project Acronym:	West-Life	
Grant agreement no.:	**675858**	
Deliverable title:	Report on prototypes constructed using Big Data approaches.	
WP No.	7	
Lead Beneficiary:	CSIC	
WP Title	Joint Research Action	
Contractual delivery date:	30 April 2018	
Actual delivery date:	30 April 2018	
WP leader:	Jose Maria Carazo	Partner CSIC
Contributing partners:	STFC, EMBL-EBI	

Deliverable written by Chris Morris STFC, Rob Firth STFC, Martyn Winn STFC, Francesco Talo EMBL-EBI

Contents

Executive summary

We have identified two projects for prototyping which are useful in the structural biology domain and also acts as a prototype for future work, demonstrating a particular Big Data technology and providing some initial useful functionality.

The first uses Natural Language Processing (NLP) methods applied to the structural biology literature to identify some information that is not currently incorporated into databases: structural annotations on specific residues within proteins. The software created by this pilot project is an open source software package, `pyresid`, written in the Python programming language. Using it, annotations will be made to all past papers with known links to entries in the Protein Data Bank. These annotations are available in EuropePMC.

For the second prototype, we looked at the use of Convolutional Neural Networks for distinguishing between protein and noise in cryoEM maps. Python scripts for generating input data from structural biology data for the machine learning, and for creating and training a model, are made available. While this serves as a useful prototype, further cleaning of the input data and better training of the model are still required.

Project objectives

With this deliverable, the project has reached or the deliverable has contributed to the following objectives:

No.	Objective	Yes	No
1	**Provide analysis solutions for the different Structural Biology approaches**		x
2	**Provide automated pipelines to handle multi-technique datasets in an integrative manner**	x	
3	**Provide integrated data management for single and multi-technique projects, based on existing e-infrastructure**		x
4	**Foster best practices, collaboration and training of end users**	x	

Detailed report on the deliverable

3.1 Background

Work Package 7 of West-Life is a Joint Research Activity aimed at exploring new ways to use existing or close to existing services so that broader user communities will be reached. One of its objectives is "Studying large sets of output data using Big Data approaches".

We first consulted partners about the most useful projects to undertake. The selected projects were then executed by partners STFC and EMBL-EBI.

3.2 Selection of projects

Milestone 30 "Big Data software introduced" described a number of Big Data technologies that could be applied to structural biology data. Subsequent discussions among partners identified the following potential projects:

- particle picking and/or conformational classification for cryoEM
- tomograph matching, with applications to EM and combined methods
- protein structure prediction, in particular torsion angle prediction
- binding site identification
- use of Natural Language Processing (NLP) methods to increase the accessibility structural publications to researchers.

These potential projects were then assessed. The criteria were:

- The selected project should add value to existing structural services
- The project should be a pilot, that tests the potential for future advances
- The project should be able to demonstrate a proof of concept using the limited resources in WP7
- The project should be novel, rather than on a topic where there is existing work

Text Mining

In the light of this, the NLP focus was selected for the first prototype. An NLP challenge is defined by the corpus of texts to be processed and the question to be answered.

The corpus was defined as all papers which are linked in EuropePMC to PDB entries, and for which full text is available through EuropePMC. It consists of 76,120 papers at the time of writing [http://europepmc.org/search?query=%28IN_EPMC:y%29+AND+%28ACCESSION_TYPE:pdb%29&page=1].

The question was defined as the identification of mentions of specific protein residues (amino acids), and linking them to the relevant protein record in UniProt. This question was specified by West-Life partner Sameer Velankar, Team Leader, Protein Data Bank in Europe (PDBe). Previously, such residue-level information has not been available in the Protein Data Bank.

The goal was defined as extracting information from all 76,120 papers in the corpus and pushing it to the EuropePMC annotation repository, and establishing a pipeline that will automatically annotate future papers.

Image processing with Neural Nets

A second prototype was selected, using machine learning to recognise features in cryoEM maps. While machine learning has been applied several times for the particle picking problem, it has not been used for identifying features of single particle reconstructions. Applications include distinguishing protein vs nucleic acid maps, identifying missing components in noisy maps, or automatic recognition of side chains as input to model building.

For the prototype, we focussed on the simple question as to whether a 2D slice from a cryoEM map contains protein or not. Although a very simple use case, it demonstrates several important features: preparation of the training dataset, effect of map blurring, choice of machine learning model architecture, and initialisation and refining of weights.

3.3 Implementation of text mining tool

3.3.1 Outcome

Figure 1 shows an example of the information that has been retrieved. This information will be uploaded to the PDB to provide annotations about specific residues. It will also be supplied to EuropePMC as annotations to all past publications that are linked to specific PDB entries.

Residue	Section	Sentence Context
Glu61	Results	These important residues are in the heme pocket, specifically in the lower cleft: Gly56, Gly57, Asn58, Ala59, Thr60, Glu61, Gln62.
Glu61	Discussion	The residues with high chemical shift perturbations in the upper and lower clefts are: Thr38 in the α2 helix, Leu51 in the α3 helix, Glu61 in the α4 helix, and Arg73 in the α5 helix.
Glu61	Discussion	Chemical shift perturbation data indicate the residues in the lower and upper clefts to be: Thr38 in the α2 helix, Val50 and Arg52 in the α3 helix, and Glu61 and Phe63 in the α4 helix.
Glu61	Discussion	The differential line broadening data mostly reveal residues from turn 3 to the α4 helix including Gly56, Gly57, Asp58, Ala59, Thr60, Glu61, and Asn62.
Glu61	Discussion	Eleven residues are indicated to be highly involved in binding: three in β3 sheet, Ile29, His32, and Tyr35; two in α2 helix, Thr38 and Lys39; three in α4 helix Thr60, Glu61, and Asp65; the axial ligand His68; Thr70 in α5 helix, and Leu84 in a loop.
Glu61	Discussion	The chemical shift perturbation data from nanodiscs does reveal more residues, with many around the turn3 to α4 helix, similarly to the bicelles reconstituted cyt b 5 data: Gly57, Asp58, Glu61, and Phe63.

Figure 1 – Example of the contextual information that can be returned by `pyresid` , in this example for mentions of Glutamate at position 61 (Glu61) in Gentry et al. 2017 (PMC5552742). Precise matches are highlighted.

In order to meet the key aim of associating the identified residues with the relevant UniProt protein record, proteins structure mentions must be identified within the text. There are two ways in which this is done within `pyresid`. One is to use the existing annotations provided by EuropePMC through their API. The second is to perform a rule-based search for Protein Data Bank identifiers (PDB IDs) within the full text. As with the search for the residue location, this is performed using regular expressions. Because the format of the PDB ID (a four character code of letters and numbers) is somewhat simpler than that of a residue, there is a greater number of false positives when searching for PDB IDs. Therefore we perform a second operation, which is to query the PDB API for a list of approved and pending IDs. Spurious matches are ruled out if they do not appear in this list.

3.3.2 – Design Rationale

Given the well-defined goals of the project, namely the size of the proposed corpus and the drive for usability for researchers, the choice of both basic approach and language choice were both key considerations. As a result, the project was written in Python, using regular expressions as its initial extraction layer.

The well constrained encoding of protein residues, with the combination of an amino acid identifier and a positional integer, lends itself to pattern matching and in particular Regular Expressions (RegEx). Pattern matching has been used in the process of text mining similar corpora (notably with *MuteX* – Horn et al. 2004, and Nagel et al. 2009, Ravikumar et al. 2012). Though these text mining frameworks include protein structure extraction, they are either hard to access and implement in a pipeline (not being open source), or fundamentally overall too slow to reasonably annotate the existing corpus, and to keep up with the growing volume. Faster Natural Language Processing (NLP) pipelines are now available, and can provide processing speeds orders of magnitude above those quoted in the aforementioned literature.

In addition to the rapid evolution of techniques in NLP, the field has moved on considerably and there is currently no bridge between the latest APIs of resources such as the EuropePMC and the PDB, and amino acid residue annotation, meaning the volume of data is difficult to leverage. The text mining tool that has been built as a result of this work provides a bridge between the various data sources, a sophisticated information extraction algorithm, and a link to cutting edge NLP pipelines. This opens up opportunities for further work.

3.3.3 – `pyresid`

The production service created by this pilot project is a software package, `pyresid`, written in the Python programming language (Python 3). Python is widely used, easily learned, and has a wealth of existing libraries and packages for NLP. Crucially, Python bindings exist for the APIs that provide the data necessary for the project. The `pyresid` package can be installed using the `pip` package manager, and the source code and documentation is available to download under a BSD 3-Clause License on the PyPi package index[1],[2]. API calls to the EuropePMC are handled for the end user by the code, so all that is needed is a PMC identifier. However, if specific queries are desired, the module also contains a suite of functions to send custom queries to the portal.

With the PMC identifier in hand, the module requests a copy of the full-text article in XML format, and parses it using the BeautifulSoup Python library[3] which is used to extract the text, section titles and other metadata from the XML. These attributes are collected together into a Source object that is used to group together the downloaded, parsed and mined data in one place. The full text of the manuscript then undergoes the initial rule-based NLP using regular expressions, extended from a set of rules from Nagel et al. 2009 (c.f. Table 1 of that work), which also provides an annotated set of Structural Biology Abstracts (the “Nagel Gold Corpus”) for algorithm evaluation. These rule extensions include a more complete treatment of residue mentions that are mentioned in complexes, separated by dashes or slashes, for example from Vigouroux et al. 2017 “*Members of the OccJ subgroup sharing >69% sequence identity*

[1] https://pypi.python.org/pypi/pyresid
[2] additional documentation can be found here: http://robfirth.github.io/pyresid/pyresid.pdf
[3] https://www.crummy.com/software/BeautifulSoup/

possess the octopine binding signature Glu30-Tyr33-Trp71-Ser91-Arg96-Gln159-Asn111-Thr163-Ala164-Asn202.". Comparisons between the Nagel Annotations and those produced by `pyresid` show a superior recall for our approach – for example, the Nagel patterns would identify the above complex mention, as a single residue.

The matches, both simple and complex, are initially converted into a `pyresid` Match Class instance, which includes the precise string that triggered the match and the indices of the start and end of the sub-string (in characters relative to the start of the full text document). A simple match, such as "*Tyr99*" is matched as:

```
{'start': 0, 'end': 5, 'string': 'Tyr99'}.
```

Whereas a more complex mention – "*Glu30-Tyr33-Trp71"* - is initially matched as

```
{'start': 0, 'end': 17, 'string': 'Glu30-Tyr33-Trp71'}.
```

A further decomposition step is performed on the complex matches to extract individual residues from the parent match. When this has been done, the parent match is discarded, but the parent string is inherited into the decomposed MatchClass. The next step is a dictionary lookup, that isolates the Amino Acid name and positional information from the match. For example, the residue mention "Tyr99" becomes a Match with the following attributes:

```
{'start': 0, 'end': 5, 'string': 'Tyr99', 'position': ['99'],
'aminoacid': 'Tyrosine', 'threeletter': 'Tyr'}.
```

While "a serine at position 97" similarly gets matched as:

```
{'start': 2, 'end': 24, 'string': 'serine at position 97', 'position':
['97'], 'aminoacid': 'Serine', 'threeletter': 'Ser'}.
```

The three-residue dashed mention above gets decomposed into three separate matches:

```
{'start': 0, 'end': 17, 'string': 'Glu30-Tyr33-Trp71', 'aminoacid':
'Glutamic acid (Glutamate)', 'threeletter': 'Glu', 'position': '30',
'residue': 'Glu30'}

{'start': 0, 'end': 18, 'string': ' Glu30-Tyr33-Trp71', 'aminoacid':
'Tyrosine', 'threeletter': 'Tyr', 'position': '33', 'residue': 'Tyr33'}

{'start': 0, 'end': 18, 'string': ' Glu30-Tyr33-Trp71', 'aminoacid':
'Tryptophan', 'threeletter': 'Trp', 'position': '71', 'residue':
'Trp71'}
```

Next, the largest NLP step is undertaken – lexical analysis, commonly known as tokenization. The tokenization is performed by using spaCy, a publicly available off-the-shelf python/cython stack, that claims to provide the fastest syntactic parser in the world[4] and has seen widespread adoption within the NLP community, including on a number of projects within life sciences[5]. By operating in an object-

[4] Choi et al. 2015, Honnibal et al. 2015
[5] see https://spacy.io/usage/resources#libraries

oriented manner with a Source Class, we are able to maintain spaCy's token-sentence-document internal structure, a feature that will be crucial for further, more sophisticated NLP.

The tokenization splits the text into sentences as well as word and punctuation tokens, based on a pre-defined language model. Each of these tokens, as well as being parsed, undergoes Part-of-Speech (POS) tagging, becoming tagged with information about its word shape and syntactic dependency. The model that is used in the current implementation of `pyresid` is an English Language model 'en_core_web_lg 2.0.0', which is based on a Convolutional Neural Network (CNN) trained using the OntoNotes 5.0 data release (Weischedel et al. 2013). The model includes amongst its data sources broadcast news, telephone and broadcast conversation transcripts and web blogs. One of the motivations in using spaCy as the NLP engine of the project was the ability to retrain and extend existing models, as it is likely that the EuropePMC literature corpus contains language that is sufficiently different to everyday usage. Errors could therefore occur in the parsing and tokenization, for example with domain-specific contractions such as species names. At present, this does not appear to be a dominant source of errors, though further work including a more comprehensive set of quantitative evaluation would illuminate the issue.

The output of the tokenisation is a spaCy document. At present, the principal use of this tokenization is to extract parent sentences, and thus extract context for the residue mentions. With a sufficiently annotated training set of sentences, it would be possible to build a specific Structural Biology model (within a spaCy pipeline) based upon the corpus. This could include a named entity recogniser that can identify residue mentions, or to formulate a classifier externally that uses vectorised context as a feature set for discrimination between relevant and irrelevant sections of text.

Once the tokenisation is complete, the previously identified matches are decorated with their associated tokens and sentence context. At the same time, the sections from the initial XML input are themselves matched into an object. As the EuropePMC uses a strict whitelist of section names for its hosted annotations, the scraped sections are matched to this whitelist using FuzzyWuzzy[6], a fuzzy matching Python module. These sections provide broader context to the discussion of residues within the manuscript, aiding end users in the performance of their research, a visualisation of the location of specific residue mentions within a manuscript can be seen in Figure 2.

[6] https://github.com/seatgeek/fuzzywuzzy

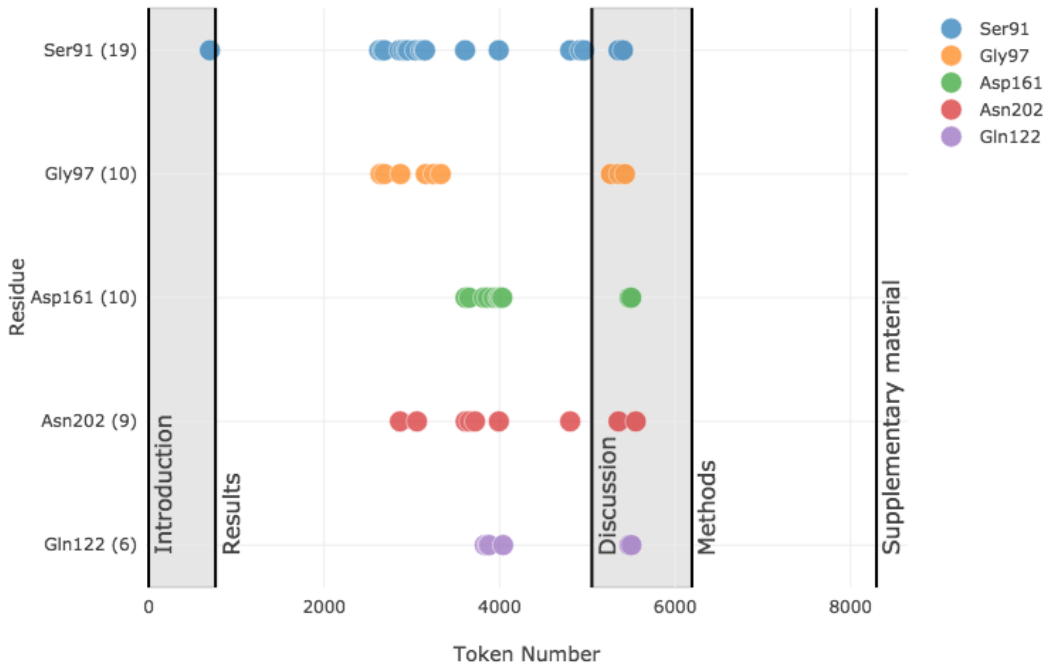


Figure 2 – A graphical representation of the locations of the 5 most common residues within Vigouroux et al. 2017 extracted by `pyresid`, and the position of the located sections within the manuscript.

As well as displaying the position within the text, and highlighting the relative position of mentions of certain residues, accurate location means that operations such as word clustering can be performed, as shown in the dendrogram in Figure 3. Functionality has been built into `pyresid` to generate all three of these visualisations; an example of a web-app using these outputs can be found here - https://pyresid-dash.herokuapp.com/.

In order to associate residues with a PDB entry, the RCSB PDB API is queried by `pyresid`, and if there is not a local copy, a .mmCIF file is downloaded. This contains the structural information for the entry, and for each residue, the structure is checked to see whether a residue of the given amino acid exists at the relevant position. If it does, the mention is annotated with the PDB ID. The individual mentions of each valid PDB ID within the text is also added to the Source object as a Match, enabling the same location plotting as for the residues (Figure 4). This location information within the text could be used to place priors on, or even replace, the structural checking with a graph-based approach (c.f. Ravikumar et al. 2012). Further work could focus on location of the protein structure by name, giving a greater number of edges to a graph-based association approach.

The number of PDB entries found by both methods is generally comparable to the number of unique residue mentions in a given manuscript. However, a common finding is that the number of unique UniProt identifiers for the protein structures is considerably smaller than the number of unique PDB entries – it being the case that the PDB entries are for substructures of a parent protein, with its own UniProt URI. As a result, mismatches between residues and protein structures occur less frequently than would be naively expected. At this point, the Match object for each mention is updated with the UniProt URI corresponding

to the PDB entry that it was formerly matched with.

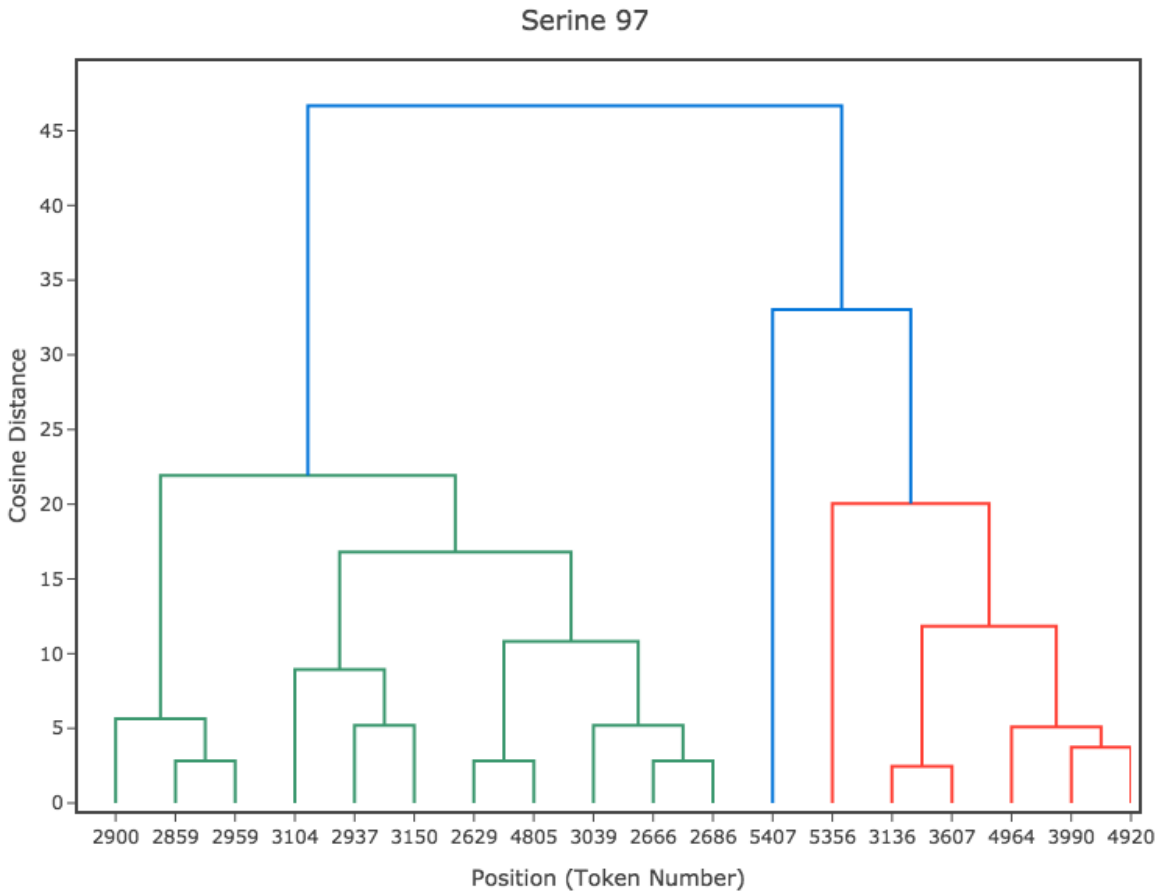


Figure 3 - A graphical representation of clusters of mentions of a Serine Residue at position 97 within Vigouroux et al. 2017 (PMC5740067).

In order for the Matches to be used as annotations, and for the maximum interoperability with other tools, at the end of processing each document, a data file in JavaScript Object Notation (JSON) file is produced, containing the annotations. Python Pickle files can also be chosen as an output, and the `pyresid` tools can be used in interactive mode in order for end users to interact more easily with the processed data.

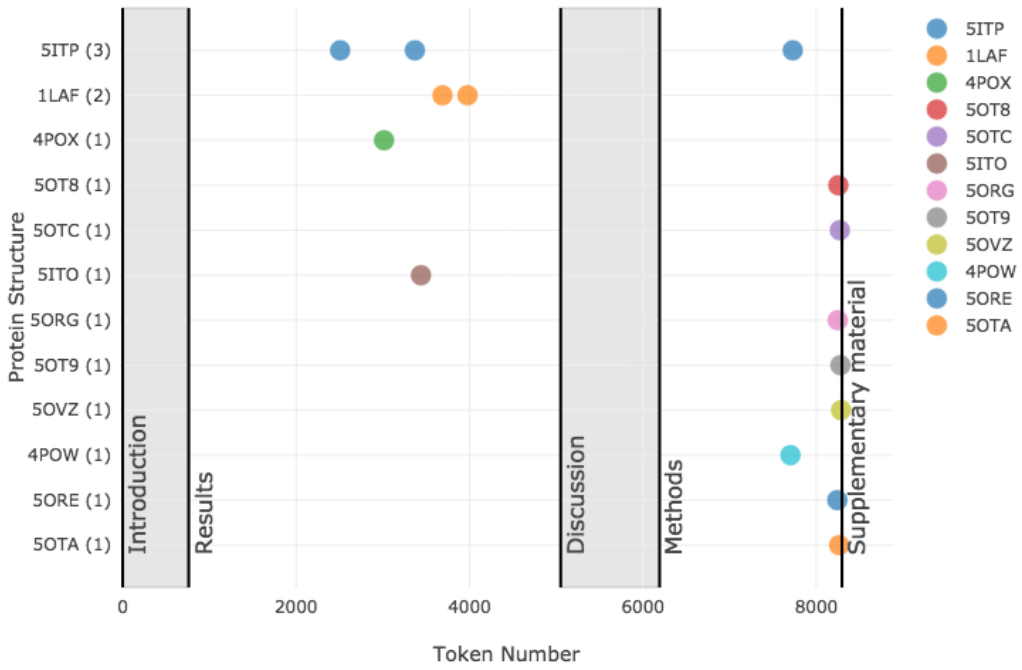


Figure 4 - A graphical representation of the locations of the Protein Structures within Vigouroux et al. 2017 (PMC5740067) extracted by `pyresid`, and the position of the located sections within the manuscript.

The annotations are now available in the development version of EuropePMC. The figure shows annotations for Vigouroux et al, “Structural basis for high specificity of octopine binding in the plant pathogen Agrobacterium tumefaciens.” This is a screenshot of the page http://dev.europepmc.org/articles/.

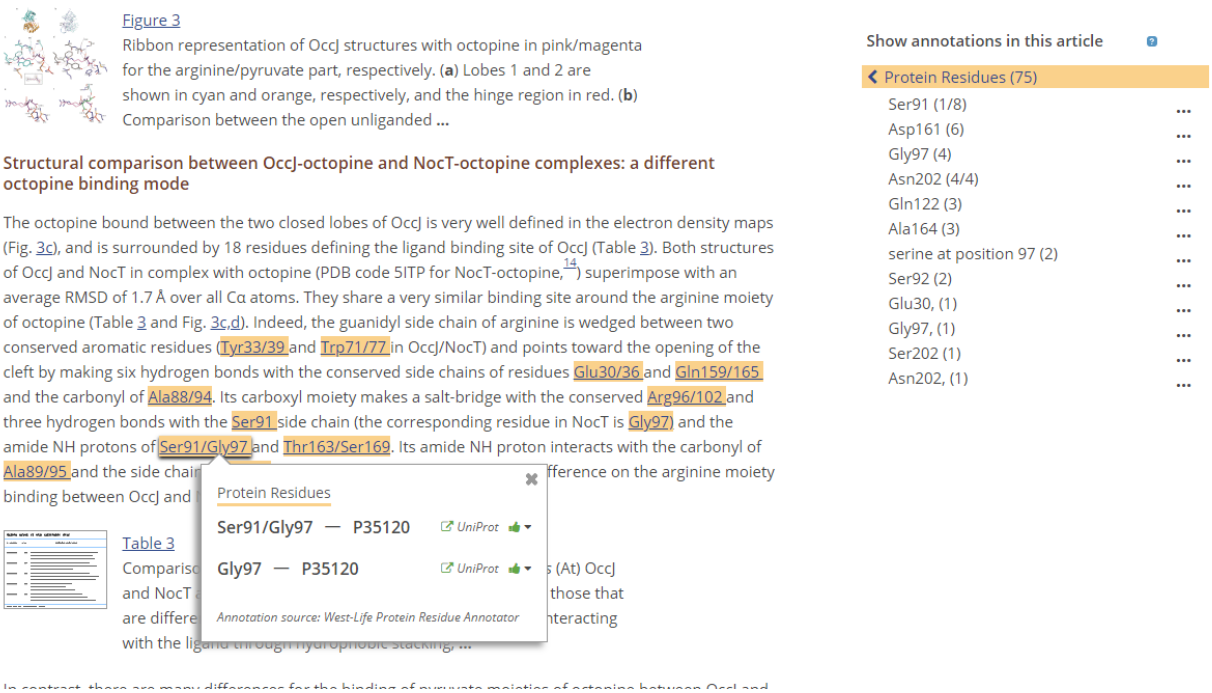


Figure 5 : Annotations for Vigouroux et al. 2017 (PMC5740067)

3.3.4 Collaborations

This work has been discussed with the International Union of Crystallography, a learned society which is the major publisher of crystallographic journals. A large majority of their publications are open access. They are considering running the West-Life annotation tool to push information about residues to EuropePMC, which would make mined information available from the non-open-access papers.
We corresponded with the OpeMinTed project about this work. They offer tools for corpus extraction and processing. Unfortunately, we needed a specialized criterion for corpus definition, as described above, namely papers linked to PDB-e entries. This selector is provided only by the Advanced Search facility of EuropePMC. A lesson from this is the value of being able to import a list of DOIs from a domain-specific search service (e.g. PDBe query system for all macromolecular structures in the PDB) as a corpus definition to a text processing service to take advantage of existing text mining facilities.

3.3.5 Future work

One challenge for this work is that NLP tools devised for standard English text work less well with scientific papers. For example life sciences papers contain many species names, e.g. "*E. coli*". The tool chain we were using interprets the abbreviation sign '.' as the end of a sentence. The Europe PMC text mining pipeline uses as number of specialist vocabularies for entity recognition such as species names (NCBI Taxonomy) and gene/protein names (UniProt Names and Synonyms) list. These vocabularies are further processed to remove highly ambiguous terms, before they are used in the daily text mining workflow, which operates on all incoming content to Europe PMC. Using such vocabularies alongside machine learning techniques that operate specifically on research papers would likely give better results. Integrating such approaches into the tokenizer would mitigate this problem. More generally, better results would be obtained by retraining the language model with an appropriate corpus. Partners STFC and EMBL-EBI will seek to define and gain funding for a future project that addresses such challenges. Enabling such follow on work was one of the aims of this work package.

3.4 Implementation of Convolutional Neural Network for structural biology maps

3.4.1 Input data

We chose as an example dataset, the 2.2Å single particle reconstruction of beta-galactosidase, deposited in EMDB as EMD-2984. This is relatively high resolution for cryoEM, and should give clear features that could potentially be recognised by a CNN. An atomic model has been built into the map (PDB 5a1a) which serves to indicate which parts of the map are really protein and which are noise (i.e. the ground truth).

The cryoEM map consists of a regular 3D grid of values, representing the electrostatic potential at each grid point. A survey of medical image analysis (Geert Litjens et al. (2017)) reveals a variety of approaches to analysing 3D datasets. Machine learning techniques are more mature for 2D images, and so many approaches consider 2D slices through a 3D dataset, or 2D projections. Nevertheless, some groups have attempted to train a model directly on 3D volumes or subvolumes.

For this project, we started with the simplest approach of extracting 2D slices from the cryoEM map. Each slice is 48 x 48 pixels. Given the pixel size of 0.637Å this corresponds to a slice of 30.6Å x 30.6Å, containing a few 10s of atoms. Smaller slices could be used to identify side chains, or larger slices to identify domains. The 48 x 48 slice is windowed across the full size of a section of the cryoEM map, using a step size of 10 pixels. Sections were selected in two ways: either every 3rd section or every 10th section. The process is repeated 3 times, taking slices perpendicular to the 3 axes.

We considered both the cryoEM as deposited in EMDB, and a blurred map (obtained using Gaussian blurring in Chimera with sd 1.0). Values in all images are normalised to the range -0.10 to 0.14 or -0.02 to 0.05 respectively, corresponding roughly to the range of values in the cryoEM maps.

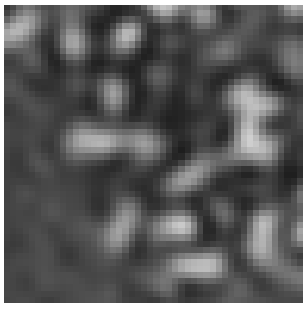
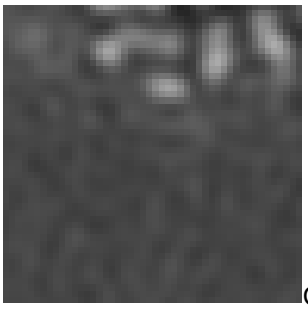
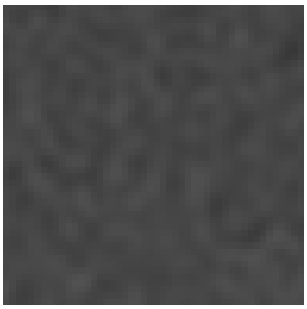
A:  B:  C: 

Figure 6: Example 2D slices taken from EMD-2984. Images A and B are classified as protein, whereas C is non-protein.

We also generated a map from the fitted model 5a1a. For each slice of the experimental map, the model map is searched for significant density. If 5% of the grid points of model density were larger than 0.1 then we considered that this is a modelled part of the map and that it corresponds to protein. In this way, we could automatically classify all slices as protein or non-protein, giving us data suitable for supervised learning. There is clearly scope to improve the annotation of training images.

3.4.2 Model training

There are many possible architectures of CNNs. To simplify things for this prototype, we decided to adopt a model which had performed well in the ImageNet Challenge, namely VGG16 from Visual Geometry Group in Oxford. The default VGG16 model is shown in Fig 7, and consists of multiple convolutional layers interspersed with max pooling layers. The final layers are fully connected, with the final layer using softmax to make the binary classification. We use a simplified version of this model, namely an input of 48 x 48 images, rather than the default 224 x 224; one convolutional layer, one pooling layer, one fully-connected layer and a softmax-based prediction layer.

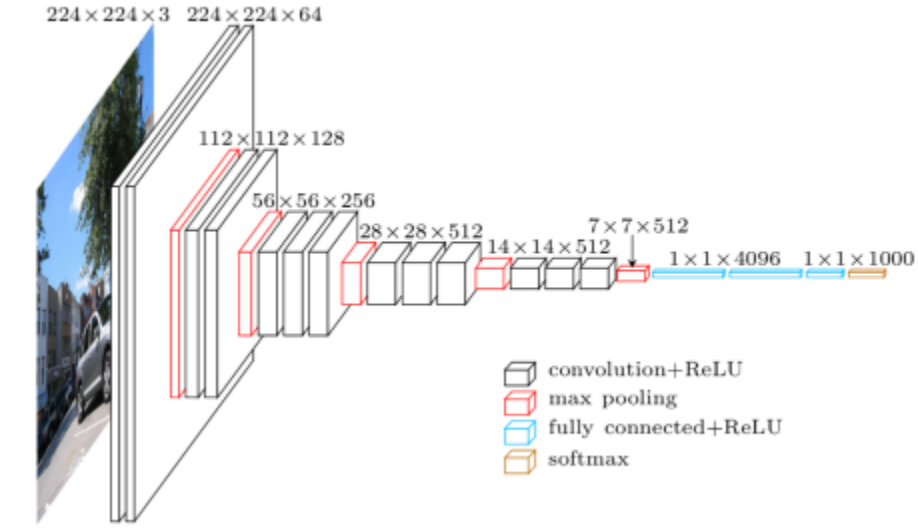


Figure 7: The VGG16 model.

In fact, early attempts to train this model failed, and we moved to a simplified version consisting of layers block1_conv1, block1_pool, flatten, fc1, predictions. Future work will look at re-introducing additional layers from VGG16, in the expectation that this will give a more robust and transferable model.

Model creation, training and testing is done in Python using the Keras package. We have used the Theano backend, although TensorFlow can also be used. The VGG16 model is available in Keras, and can be imported with:

```
vgg_model = vgg16_mdw.VGG16(weights=None,
                            input_shape=input_shape,
                            include_top=True,
                            classes=1)
```

As explained above, we have used a subset of layers from VGG16, which are added individually to the model with the appropriate model.add() command in Keras. We have not tried to use the available pre-trained weights, but have attempted to train the model using our structural biology derived images.

3.4.3 Results

The first dataset used for training consisted of slices taken from every 3rd section on a single axis of the blurred cryoEM map. There are more slices classed as non-protein, and so these were trimmed to give a balanced dataset of 15,696 protein images and 17,452 non-protein images (47% protein: 53% non-protein images). This set was shuffled, and split into 80% training and 20% test images.

Attempts to train the reduced 5-layer model with all images failed. Several weight initialization schemes were tried. However, training on just 2 images and slowly adding in more images worked, allowing the weights to refine sensibly. The final model had a reported loss of 0.099 and an accuracy of 0.978.

We next considered a dataset from the original (unblurred) cryoEM map. Taking slices from every 10th section on a single axis yielded 4,736 protein images and 14,014 non-protein images. Using the previously obtained model, and not refining the weights, gave a loss of 11.842 and an accuracy of 0.252. In other words, a model trained on blurred images is not predictive for images taken from an unblurred, noisy map.

No. images	**Protein**	**Non-protein**	**Loss**	**Accuracy**
Model trained and tested on images from blurred map				
33,148	15,696	17,452	0.099	0.978
Model trained on blurred map, tested on images from unblurred map				
18,750	4,736	14,014	11.842	0.252
Model trained and tested on images from unblurred map				
18,750	4,736	14,014	0.176	0.938
Model trained on unblurred map, tested on images from blurred map				
33,148	15,696	17,452	1.631	0.587

Table 1: Summary of testing of machine learning model

Training the model on images from the unblurred map, starting from the previously obtained weights, led to a much improved accuracy of 0.938. Using these weights to make predictions for the original blurred map again gave poor results (loss of 1.631 and accuracy of 0.587).

In conclusion, we are able to train a 5-layer model to make good predictions for either the original cryoEM map, or for a blurred map. However, cross predictions are poor.

3.4.4 Future work

Further work is needed to refine the network, both in terms of architecture and preparation of input data. The aim of course is to have a single model which is predictive for a range of cryoEM maps, with different levels of blurring or sharpening. This will likely require going to a more complex model, for example the full VGG16 network. Although our original attempt at VGG16 failed, it should be possible to bootstrap up from our 5-layer model, adding additional layers in a gradual process.
The example chosen is very simple – identifying regions of a map which can contain protein. A more challenging, but more useful, problem would be to distinguish protein-like density from nucleic acid-like density. The same approach can be taken, using e.g. a modelled ribosome structure to provide training data.

3.4.5 Availability

The python scripts used to extract 2D images from cryoEM maps, and to train the model using Keras, will be made available via https://github.com/martynwinn/map-recognition. Manipulation of cryoEM maps is performed using the CCPEM library mrcfile (https://pypi.org/project/mrcfile/), which reads maps downloaded e.g. from EMDB and converts the data into numpy arrays. The site will also contain model architectures as JSON and YAML files, and trained weights in HDF5 format. These are all suitable for loading into Keras.

References cited

Choi, J.D., Tetreault, J.R., & Stent, A. (2015). It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool. *ACL*.

Florence Horn, Anthony L. Lau, Fred E. Cohen; Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors, *Bioinformatics*, Volume 20, Issue 4, 1 March 2004, Pages 557–568, https://doi.org/10.1093/bioinformatics/btg449

Gentry KA, Prade E, Barnaba C, Zhang M, Mahajan M, Im SC, Anantharamaiah GM, Nagao S, Waskell L, Ramamoorthy A. Kinetic and Structural Characterization of the Effects of Membrane on the Complex of Cytochrome b 5 and Cytochrome c. Sci Rep. 2017 Aug;7(1) 7793. http://doi.org/10.1038/s41598-017-08130-7. PMID: 28798301; PMCID: PMC5552742.

Honnibal, M., & Johnson, M. (2015). An Improved Non-monotonic Transition System for Dependency Parsing. *EMNLP*.

Geert Litjens et al. (2017) "A Survey on Deep Learning in Medical Image Analysis" arXiv:1702.05747

Nagel, K., Jimeno-Yepes, A., & Rebholz-Schuhmann, D. (2009). Annotation of protein residues based on a literature analysis: cross-validation against UniProtKb. *BMC Bioinformatics*, *10*(Suppl 8), S4. http://doi.org/10.1186/1471-2105-10-S8-S4

Ravikumar, K., Liu, H., Cohn, J. D., Wall, M. E., & Verspoor, K. (2012). Literature mining of protein-residue associations with graph rules learned through distant supervision. *Journal of Biomedical Semantics*, *3*(Suppl 3), S2. http://doi.org/10.1186/2041-1480-3-S3-S2

Vigouroux A, El Sahili A, Lang J, Aumont-Nicaise M, Dessaux Y, Faure D, Moréra S. Structural basis for high specificity of octopine binding in the plant pathogen Agrobacterium tumefaciens. *Sci Rep.* 2017 Dec;7(1) 18033. http://doi.org/10.1038/s41598-017-18243-8 PMID: 29269740; PMCID: PMC5740067.

Weischedel, Ralph, et al. OntoNotes Release 5.0 LDC2013T19. *ISBN: 1-58563-659-2*, Web Download. Philadelphia: Linguistic Data Consortium, 2013.