Supplementary Material for 'PEDL: Extracting protein-protein associations using deep language models and distant supervision.'

Leon Weber, Kirsten Thobe, Oscar Arturo Migueles Lozano, Jana Wolf and Ulf Leser

1 Presence of P53 interactions in data bases

To give an indication of the completeness of pathway data bases, we manually check whether PPAs from a state-of-the-art model of P53 signalling (Hat *et al.*, 2016) can be found in PID or Reactome. Three experts in systems biology manually converted the reactions of the model to the considered PPAs. Then we manually checked whether this PPA could be found in the PathwayCommons versions of Reactome and PID. The results of this analysis can be found in Table SM1. The model is comprised of 24 PPAs. Out of these, 6 can't be found in either data base and 12 are missing from at least one.

2 Hyperparameter settings

2.1 PEDL

We set the learning rate to 3e−5 and the batch size to 16 across all experiments. The maximum sequence length per text span is set to 256 WordPiece-tokens and all spans that exceed this length after tokenization are discarded. We implemented PEDL with the *PyTorch*[1] deep learning framework and the *HuggingFace's transformers* library[2]. To allow the model to fit in GPU memory, we represent parts of the models with only 16-bit floating point numbers (instead of 32-bit) using the *apex*-libary[3] with optimization level *O1*. On all data sets, we found $\alpha = 0.2$ to be optimal for the development data.

2.2 comb-dist

For BioNLP (*E2* & *E3*), we found *sent_loss_weight* = 1 optimal, while for PID (*E1*), *sent_loss_weight* = 2 yielded the best results. Dropout was turned off

[1] `https://pytorch.org`
[2] `https://github.com/huggingface/transformers`
[3] `https://github.com/NVIDIA/apex`

PPA	PID	Reactome
ATM *controls-phosphorylation-of* TP53	✓	✓
ATM *controls-phosphorylation-of* MDM2	✓	✓
ATM *controls-phosphorylation-of* SIAH1	✗	✗
SIAH1 *controls-state-change-of* HIPK2	✗	✗
HIPK2 *controls-phosphorylation-of* TP53	✓	✗
TP53 *controls-expression-of* MDM2	✓	✓
TP53 *controls-expression-of* PPM1D	✗	✗
TP53 *controls-expression-of* CDKN1A	✓	✓
TP53 *controls-expression-of* BAX	✓	✓
PPM1D *controls-phosphorylation-of* ATM	✗	✗
PPM1D *controls-phosphorylation-of* MDM2	✓	✗
PPM1D *controls-phosphorylation-of* TP53	✗	✗
AKT1 *controls-phosphorylation-of* MDM2	✓	✓
MDM2 *controls-state-change-of* TP53	✓	✓
CDKN1A *in-complex-with* CCNE1	✗	✓
CCNE1 *controls-phosphorylation-of* RB1	✓	✗
RB1 *in-complex-with* E2F1	✓	✓
E2F1 *controls-expression-of* CCNE1	✓	✓
YWHAZ *in-complex-with* BAD	✓	✓
AKT1 *controls-phosphorylation-of* BAD	✓	✓
BAD *in-complex-with* BCL2L1	✓	✓
BAX *in-complex-with* BCL2L1	✗	✗
TP53 *controls-expression-of* PTEN	✓	✗
MDM2 *controls-state-change-of* HIPK2	✓	✗

Table 1: Presence of the PPAs from a state-of-the-art model of P53 signalling in pathway data bases. ✓ indicates that the PPA could be found and ✗ that it couldn't.

across all experiments because this yielded the best scores on all development sets. The remaining hyperparameters were left at their default values from the original implementation.[4]

3 Transforming BioNLP data to PPA format

BioNLP Event Type	PPA
Gene_expression	controls-expression-of
Translation	controls-expression-of
Transcription	controls-expression-of
Transport	controls-transport-of, controls-state-change-of
Localization	controls-transport-of, controls-state-change-of
Phosphorylation	controls-phosphorylation-of, controls-state-change-of
Dephosphorylation	controls-phosphorylation-of, controls-state-change-of
Acetylation	controls-state-change-of
Deacetylation	controls-state-change-of
Ubiquitination	controls-state-change-of
Deubiquitination	controls-state-change-of
Hydroxylation	controls-state-change-of
Dehydroxylation	controls-state-change-of
Methylation	controls-state-change-of
Demethylation	controls-state-change-of
Glycosylation	controls-state-change-of
Deglycosylation	controls-state-change-of
Protein_modification	controls-state-change-of
Binding	in-complex-with
Dissociation	in-complex-with

Table 2: The mapping from BioNLP event types to PPAs used in the transformation of the BioNLP data. Comma-separated PPAs imply that the event type maps to more than one PPA and thus multiple PPAs are inferred for one event with this type.

The BioNLP event extraction data is distributed in the BRAT-standoff format, which was designed to annotate the textual description of complex biochemical events. Each event is defined by a trigger-word such as 'phosphorylation' and a theme which denotes the protein undergoing the change expressed by the event. Events can optionally be regulated by other proteins, which can be expressed by *cause* annotations or connected *regulation* events. We extract a PPA between proteins A and B if there is an event with theme B and regulator A. The mapping from event types to PPAs can be found in Table SM2 and we

[4] `https://github.com/allenai/comb_dist_direct_relex`

BioNLP Event Type	PPA
Binding	in-complex-with
Catalysis of acetylation	controls-state-change-of
Catalysis of glycosylation	controls-state-change-of
Catalysis of hydroxylation	controls-state-change-of
Catalysis of methylation	controls-state-change-of
Catalysis of phosphorylation	controls-phosphorylation-of, controls-state-change-of
Catalysis of ubiquitination	controls-state-change-of
Regulation of expression	controls-expression-of
Regulation of phosphorylation	controls-phosphorylation-of, controls-state-change-of
Regulation of localization	controls-transport-of, controls-state-change-of
Regulation of transcription	controls-expression-of

Table 3: The mapping from EVEX relation types to PPAs used in the transformation of the BioNLP data. Comma-separated PPAs imply that the EVEX type maps to more than one PPA and thus multiple PPAs are inferred for one EVEX relation with this type.

discard all events that cannot be mapped to a PPA. If an event type maps to multiple PPA-types, we infer multiple PPAs accordingly.

4 Transforming EVEX data to PPA format

We transform the binary EVEX into our PPA format by applying the type mapping given in Table SM3. The proteins in EVEX are identified by Entrez ids, which we map to Uniprot identifiers by querying MyGeneInfo for the top-scoring human Uniprot id. If an EVEX type maps to multiple PPA-types, we infer multiple PPAs accordingly.

5 Annotation guidelines

The goal of this analysis is to evaluate how well the compared models perform in evidence prediction. A text span is called evidence for a given PPA between two proteins if the relation between both proteins is asserted somewhere in it. For instance, if the given PPA is 'BTC *in-complex-with* ErbB4', then a supporting text span for the PPA would be 'BTC is a ligand of ErbB4'. A statement that would not be considered as support would be 'We estimate the expression of BTC and ErbB4'.

Biological background knowledge is disregarded and only explictly stated facts are considered. If a sentence reads 'MAPK and its substrate MAP...', then this does *not* express a phosphorylation PPA, even though you know that MAPK is a kinase.

One mention per protein is marked in the text span. If the relation is expressed between different mentions of the same proteins then the text span still counts as evidence. The entity marking is only given for your convenience.

Sometimes the protein normalization (mapping from protein mention to database id) might be wrong. In those cases, if the text span expresses the relation between wrongly normalized proteins, this still counts as evidence, because we are not evaluating a normalization method.

If it is ambiguous whether an expressed relation is direct or indirect, the text span should be counted as evidence. If the stated relation is clearly indirect, the text span shouldn't be counted as evidence. For instance, 'MEKK2 activates ERK5' should be considered as evidence, whereas 'MEKK2 activates ERK5 via MKK5' should not.

Complexes are frequently written down as 'ProteinA/ProteinB'. However, it is frequently unclear whether this denotes a protein complex or means 'ProteinA or ProteinB'. If it is not obvious that this refers to a complex, the text span should not be counted as evidence. For instance, 'the dimer ProteinA/Protein' should be considered as evidence, whereas 'We study ProteinA/ProteinB', shouldn't.

We provide a browser-based viewer for the predictions of the model. A screenshot can be found in Figure SM1

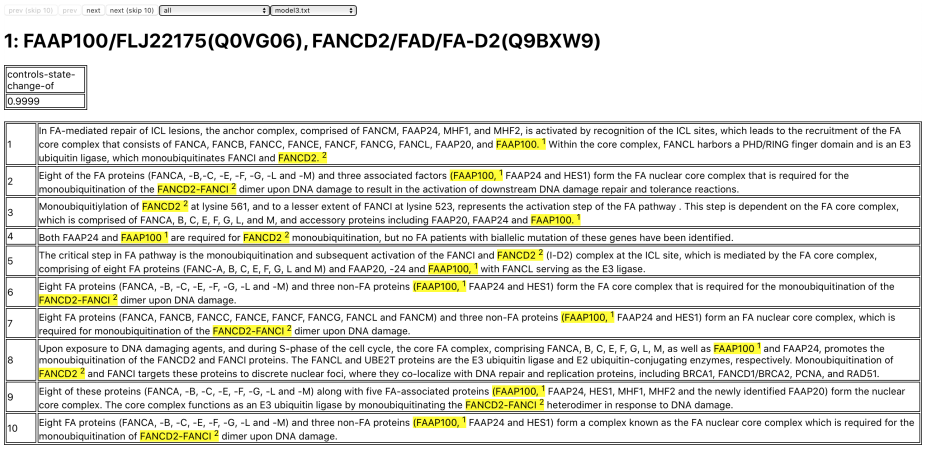

Figure 1: The browser-based relation viewer used for annotation.

6 Comparison to comb-dist

In this section, we provide a detailed comparison of PEDL to comb-dist. Both PEDL and comb-dist are multi-instance learning methods developed for distantly supervised relation extraction. They both allow for inclusion of directly supervised data to improve the accuracy of the model.

However, there are large differences with respect to the technical details of

both methods. comb-dist employs a Piece-wise Convolutional Neural Network (PCNN) with Selective Attention and Word Embeddings as its machine learning model. First, each word is encoded with pretrained (uncontextualized) word embeddings. The PCNN first divides the text span into three subspans: (1) the text left to the first entity, (2) the text between both entities and (3) the text right to the entity. Each of these three subspans is then processed by a Convolutional Neural Network (CNN) layer that models interactions between words that are closely together. The resulting representations are then aggregated with max-pooling, yielding a vector representation of each text span. Finally, the vector representations of each text span are aggregated with selective attention and the resulting encoding of all text spans is used for prediction.

PEDL on the other hand uses pretrained transformers which recently led to large gains over CNN models in many other NLP tasks (Devlin et al., 2019; Beltagy et al., 2019b). In PEDL, each word in the text span is encoded by the self-attention mechanism of the transformer, allowing to model interactions between words regardless of their respective distance. The weights of the transformer have been pretrained with masked language modelling and next-sentence prediction on a corpus that includes a large number of biomedical articles. This allows the model to learn (general and domain-specific) regularities of language which are then encoded in the contextualized word embeddings. Each text span is then represented by the embedding of a special token, that was pretrained to encode information about the entire span. Each of the span representations is then transformed into a score vector that models which PPAs (if any) are expressed in the text span. These score vectors are then aggregated by taking the maximum or its LogSumExp approximation. Note, that no attention module is used in the aggregation of the span representations, which allows to directly interpret the score vectors as PPA-predictions per span.

7 Error analysis

We perform an error analysis on the PID test data (E1) to gain a more detailed picture of the performance differences between PEDL and comb-dist.

First, we analyze the AP as a function of the maximum number per texts. For this, we select the subset of pairs that contain at most i text spans for $i \in [1, 100)$. Then, we compute the AP of both PEDL and comb-dist for each subset. The result can be found in Figure SM2. We find that for both models performance improves with the number of available text spans, with PEDL performing roughly 5 pp better for all i's.

In a second analysis, we proceed similarly, but analyze the AP as a function of the maximum average length of texts. That is, we select the subset of all protein-pairs whose text spans have an average length of not more than i for $i \in [1, 1000]$. The results can be found in Figure SM3. For both models, there is a trend of increasing performance with a larger average text length up to the point of $i = 300$. After 300, the performance decreases slightly with growing i. PEDL performs better than comb-dist for almost all average text lengths with

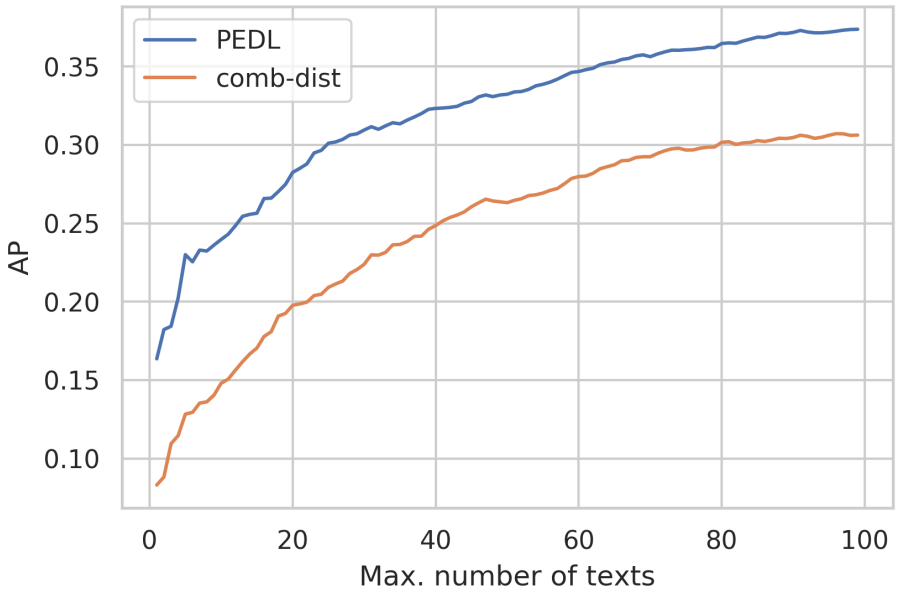

Figure 2: Average Precision as a function of the maximum number of texts. At each point, we compute the AP for the subset of bags with at most i text spans for $i \in [1, 100)$. PEDL performs consistently better than comb-dist for all maximum text numbers. The performance of both models consistently increases with the number of available text spans.

a larger difference for smaller values.

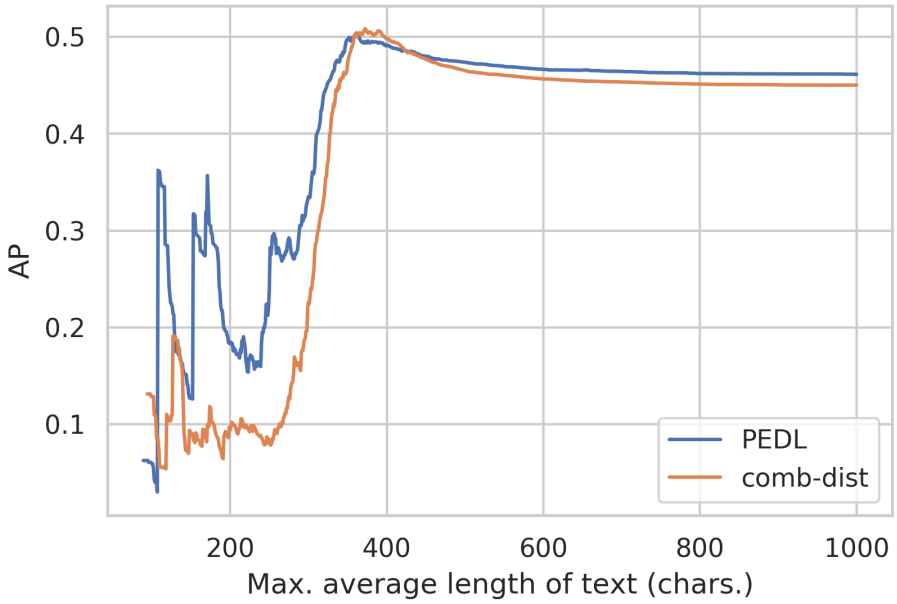

Figure 3: Average Precision as a function of the maximum average text length. At each point, we compute the AP for the subsets of bags with an average text span length of at most i for $i \in [1, 1000]$. PEDL performs better than comb-dist for most maximum average lengths with the improvement being higher for lower lengths.