AxCell: Automatic Extraction of Results from Machine Learning Papers

Marcin Kardas[1] **Piotr Czapla**[2] **Pontus Stenetorp**[3]

Sebastian Ruder[4] **Sebastian Riedel**[1,3] **Ross Taylor**[1] **Robert Stojnic**[1]

[1]Facebook AI Research (FAIR), London, UK
[2]n-waves, Wrocław, Poland
[3]Department of Computer Science, University College London (UCL), UK
[4]DeepMind, London, UK

Abstract

Tracking progress in machine learning has become increasingly difficult with the recent explosion in the number of papers. In this paper, we present AxCell, an automatic machine learning pipeline for extracting results from papers. AxCell uses several novel components, including a table segmentation subtask, to learn relevant structural knowledge that aids extraction. When compared with existing methods, our approach significantly improves the state of the art for results extraction. We also release a structured, annotated dataset for training models for results extraction, and a dataset for evaluating the performance of models on this task. Lastly, we show the viability of our approach enables it to be used for semi-automated results extraction in production, suggesting our improvements make this task practically viable for the first time. Code is available on GitHub.[1]

1 Introduction

Machine learning studies how machines learn with respect to a task, a performance metric, and a dataset (Mitchell, 2006). The (task, dataset, metric name, metric value) tuple can therefore be seen as representing a single result of a machine learning paper. To make progress as a field we need to make comparisons between results achieved with different methodologies. In light of the explosion in the number of machine learning publications in recent years, such comparisons have become more difficult.[2] This poses serious challenges to peer review, among others. For instance, across ten language modelling papers submitted to ICLR 2018, the perplexity score of the best baseline differed by more than 50 points (Ruder, 2018).

One way to deal with the deluge of papers is to develop automatic approaches for extracting results from papers and aggregating them into leaderboards. Authors typically publish their results in a tabular format in the paper, including a selection of comparisons between their approach and past papers. Automatic extraction of result tuples from tables—and optionally metadata such as model names—enables a full comparison between published methods.

Online leaderboards for comparison have become increasingly common in the research community. But these are only available for a few tasks and do not aid the comparison of models across tasks. To fill the gap, result aggregation tools such as Papers With Code[3] and NLP-Progress[4] utilise crowdsourced community contributions to populate paper leaderboards. However, human annotation of results can be laborious and error-prone, leading to omission or misreporting of paper results. This motivates the need for a machine learning approach to create a comprehensive results resource for the field.

Existing state-of-the-art approaches for results extraction are brittle and noisy, relying on text formatting hints and tables extraction from PDF files (Hou et al., 2019). In contrast, we propose AxCell, a pipeline for automatic extraction of results from machine learning papers. AxCell breaks down the results extraction task into several subtasks including table type classification, table semantic segmentation and linking results to leaderboards. We employ an ULMFiT-based classifier architec-

[1]`https://github.com/paperswithcode/axcell`

[2]In 2019, over 33,000 machine learning papers were published on the arXiv.org open-access e-print archive, with a year-on-year growth of around 50% since 2015.

[3]`https://www.paperswithcode.com/sota`

[4]`http://nlpprogress.com/`

ture (Howard and Ruder, 2018) to make full use of paper and table context to interpret tabular content, and extract results accordingly.

As a whole, this paper makes three main contributions to the literature. First, we significantly improve over the state-of-the-art for results extraction with our AxCell model. On the subset of the NLP-TDMS dataset of Hou et al. (2019) where LaTeX code is available, our approach achieves a micro F_1 score of 25.8 compared to the state of the art of 7.5. Secondly, we release a structured, annotated dataset for training models for results extraction, and an evaluation dataset for evaluating the performance of models on this task. Lastly, our approach is used in an in-production setting at paperswithcode.com to semi-automatically (by aiding the human review) extract results from papers and track progress in machine learning.

2 Related Work

Results Extraction. Previous works have studied the problem of extracting results tuples (task, dataset, metric name, metric value) from papers. Singh et al. (2019) perform search over publications and compose a leaderboard for a queried triplet. Similar to our approach, they use tables extracted from LaTeX sources. In contrast, they do not extract absolute metric values but rank papers and do not appear to utilise the text content of publications. Our goal in this paper is to extract complete results to create leaderboards, so unlike Singh et al. (2019), we focus on extracting raw metric values. Additionally we make use of the content of the publication as context for entity recognition and linking.

Closer to our formulation, Hou et al. (2019) extract absolute metric values alongside the metric name, task and dataset. They also use text excerpts as well as direct tabular information to make inferences for table contents. They frame extraction as a natural language inference problem and apply an NLI model based on a BERT architecture (Devlin et al., 2018) to extract results from PDF files. The disadvantage of this approach is that using PDFs leads to a lot of noise in structural information such as the partition of a table into cells. In our work, we explicitly utilise the structural information from the LaTeX source to extract entire tables in order to perform semantic segmentation. We demonstrate that this structural information and segmentation are crucial for boosting extraction performance.

Table Extraction. The more general problem of retrieving information from tables has been studied in past works (Milosevic et al., 2019; Ghasemi-Gol and Szekely, 2018; Wei et al., 2006; Herzig et al., 2020). Our focus in this paper is on the problem of extracting and interpreting content of tables characteristic to machine learning papers. The goal of our table semantic segmentation model is to classify cells into categories. That is, instead of performing structural segmentation when one tries to distinguish between captions, headers and rows in a stream of text (as in (Pinto et al., 2003)) we focus on semantic segmentation (i.e., assigning roles to each cell) of tables.

3 Our Approach

The task of paper results extraction is to take a machine learning paper as an input and extract results contained within the paper, specifically tuples of the form (task, dataset, metric name, metric value). As an example, if we were to take in the EfficientNet paper of Tan and Le (2019) as an input, some example results tuples we would want to extract would be EfficientNet-B7 (Image Classification, ImageNet, Top 1 Accuracy, 0.844), EfficientNet-B7 (Image Classification, ImageNet, Top 5 Accuracy, 0.971) and EfficientNet (Image Classification, Stanford Cars, Accuracy, 0.947).

To tackle this problem effectively, we need to frame the problem by defining subtasks to solve that take us from paper to results. AxCell solves several subtasks: (i) **table type classification**, identifying whether a table in a paper has relevant results; (ii) **table segmentation**, segmenting and classifying table cells according to whether they hold metrics, datasets, models, etc.; and (iii) **linking results to leaderboards**, taking the result tuples and matching them to an existing leaderboard of results. The end-to-end system is shown in Figure 1 with reference to an example. We now introduce the different components of AxCell.

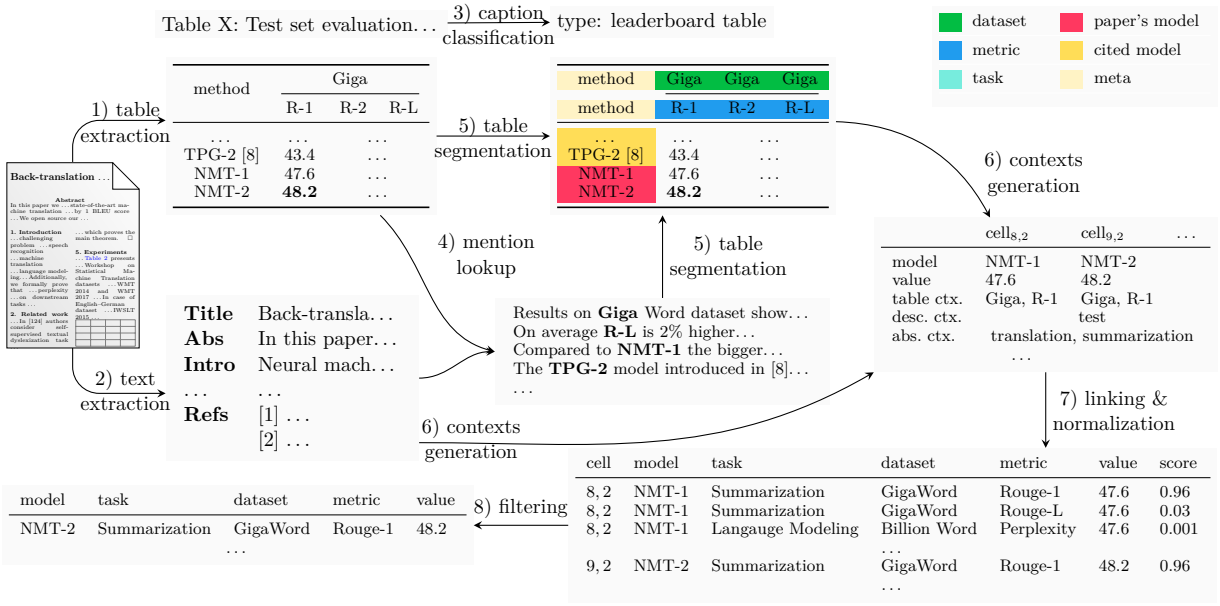


Figure 1: Graphical depiction of AxCell. The extraction starts with LaTeX source code of a paper, from which we extract 1) tables and 2) text. 3) We classify the caption to filter out irrelevant tables. 4) The content of each cell is looked up in the paper's text. Retrieved mentions are used to 5) segment cells based on their meaning (see the legend in the top-right corner). The segmented table and the paper's text are used to 6) obtain contexts for each numeric cell. 7) Results tuples are scored based on contexts and numeric values are normalized to match required format. Finally, inferior results or results below a confidence threshold are 8) filtered out.

3.1 Table Type Classification

The first stage of AxCell is to categorize tables from papers into one of three categories: `leaderboard` tables, `ablation` tables and `irrelevant` tables. A `leaderboard` table contains the principal results of the paper on a selected benchmark, including comparisons with other papers. An `ablation` table compares different permutations of the paper's methodology. Lastly, `irrelevant` tables include hyperparameters, dataset statistics and other information that is not directly relevant for result extraction.

For this stage we employ a classifier with a ULMFiT architecture (Howard and Ruder, 2018) with LSTM layers and a SentencePiece unigram model (Kudo, 2018) for tokenization.[5] We train the SentencePiece model and pretrain a left-to-right ULMFiT language model on text of papers from an unlabelled dataset of arXiv articles (see Section 4). Table 5 in the Appendix contains details on the hyperparameters and training regime. [6]

The classifier head is a standard ULMFiT classifier with a pooling layer followed by two linear layers. We treat the problem as a two-label classification with labels: `leaderboard` and `ablation`. A table is considered irrelevant if it is neither a leaderboard nor ablation (we use a confidence threshold of 0.5). We train the model on the SegmentedTables dataset (see Section 4.2).

3.2 Table Segmentation

The second stage of AxCell is to pass relevant tables to a table segmentation subtask. The goal is to annotate each cell of a table with a label denoting what type of data a given cell contains. To this end, we classify each table cell into one of: `dataset`, `metric`, `paper model`, `cited model`, and `other` (containing meta and task cells). An example segmented table is shown in Figure 1.

To help classify each table cell, we provide a context in which the cell content is mentioned. We search for cell content in the full

[5] Our classifier uses the fast.ai implementation (Howard and Gugger, 2020).

[6] We experimented with finetuning alternative language models such as BERT and SciBERT but our initial experiments did not yield superior results. A full investigation of alternative models, including pretraining from scratch, is left for future research.

`On TREC-6, <MASK> significantly improves upon training from scratch; as examples are shorter and fewer, supervised and semi-supervised <MASK> achieve similar results.`

Figure 2: An example of a text excerpt from the paper by Howard and Ruder (2018) used as evidence for a cell content query with *ULMFiT* (covered with `<MASK>` token) as `paper model`.

paper content using a BM25 scoring algorithm. Retrieved text fragments are then passed to a ULMFiT-based classifier with some handcrafted features for the cell. These features include information such as the position of the cell in the table, whether the cell is a header, and cell styles. A full list is available in the Appendix. For processing the retrieved text fragments, the retrieved term from the cell is replaced with a special mask `<MASK>` token to inhibit memorization of common names (see Figure 2 for an example). Table segmentation can then be treated as a classification problem with 5 exclusive labels. We use the same pre-trained language model weights and SentencePiece model as for the table type classification. Results for this stage of the model are outlined in Table 3.

3.3 Cell Context Generation

The next stage after table segmentation is to generate contexts for numeric cells. As an example, if we know a numeric cell has a dataset cell somewhere in its row, and a model cell somewhere in its column, then this table context is informative for deciding the dataset and model for this result. But there is much broader context in the paper that is useful for linking.

For example, a paper studying semantic segmentation with models evaluated on KITTI and CamVid datasets could mention *semantic segmentation* in the introduction, *test set* in a subsection referring to a results table, *KITTI* in the description of that table and *class IoU* in the column header. Figure 3 shows a visual representation of this hierarchy of context.

To reflect this hierarchy we generate several types of contexts for each cell. The `table context`, as discussed, looks at a numeric cell and other cells in its row or column labeled as model, dataset or metric. We also define text

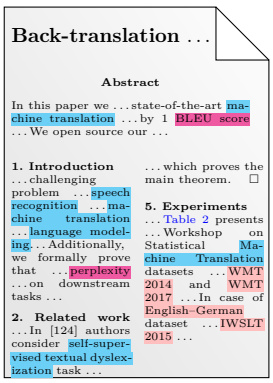


Table I: . . . test set . . . BLEU metric.

	WMT 2014		. . .
. . .	en-fr	fr-en	. . .
. . .	. . .	. . .	. . .
NMT (ours)	**56.3**	41.8	. . .

Linking result:
Task: Machine Translation
Dataset: WMT2014 English–French Test
Metric: BLEU score
Value: 56.3
Model: NMT
Confidence: 0.98

Figure 3: Using Context Hierarchy and Evidences for Linking. This figure highlights the context hierarchy, from the global paper to the specific table, the evidence for tasks (blue), datasets (pink) and metrics (violet) for the 56.3 value extracted from cell contexts, and lastly the result from linking.

contexts: a `caption context`, the table caption; a `mentions context`, text fragments referencing the table; an `abstract context`, the paper abstract; and a global `paper context`, containing the entire paper text. The gathered contexts are then used to link potential results to predefined leaderboards of results.

3.4 Linking Cells to Leaderboards

Once we have the cell contexts, the next stage of AxCell is to link them to leaderboards to form performance records. The goal is to take a metric value for a model and infer the leaderboard it is connected to. A leaderboard is defined via the triplet (task, dataset, metric name). For example: (Image Classification, ImageNet, Top 1 Accuracy) can capture papers that report performance on Image Classification for ImageNet and report Top 1 Accuracy. To simplify the problem, we assume a closed-domain with all leaderboards known in advance. To match results to leaderboards we look for evidence in cell contexts, which we now explain.

3.4.1 Inference From Evidences

Pieces of evidence are words or phrases that correspond to a task, dataset or metric. For example, *SST-2*, *binary* and *polarity* could all serve as evidence for the two-class *Stanford Sentiment Treebank* dataset (Socher et al., 2013). Pieces of evidence allow us to infer whether an entity has been mentioned in a given context. Using the same example, if "SST-2" appears in the table caption then this is evidence that

a numeric value in the table could be linked to the *Stanford Sentiment Treebank* dataset.

For a given numeric cell in the table, we search the cell contexts for evidence for every entity (task, dataset, metric) and accumulate them into a set of M pieces of cell evidence $E = \{e_1, \ldots, e_M\}$, with e_j of the form $(mention_j, entity_j, context_j)$. For example, (acc, metric, table) means "acc" metric evidence was found in cell's table context. Using this evidence set, our goal is to calculate $\mathbb{P}(y_k \mid E)$, where y_k is a binary variable denoting whether the cell contains results for a leaderboard $k \in \{1, \ldots, N\}$.

Through Bayes' Rule we know that $\mathbb{P}(y_k \mid E) \propto \mathbb{P}(E \mid y_k)\,\mathbb{P}(y_k)$. We can estimate $\mathbb{P}(E \mid y_k)$ by Naive Bayes:

$$\mathbb{P}(E \mid y_k) \approx \prod_j^M \mathbb{P}(e_j \mid y_k)$$

Since $e_j = (mention_j, entity_j, context_j)$, to model $\mathbb{P}(e_j \mid y_k)$ we need to define a joint probability model for these different elements. In our results linking model, we assume a mention can appear in a given context on purpose, to describe content of the cell, or it can be noise – falsely matched or referencing another cell. With additional simplifications we assume:

$$\begin{aligned}\mathbb{P}(e_j \mid y_k) &= \mathbb{P}\Big((mention_j, entity_j, ctx_j) \mid y_k\Big) \\ &\propto \mathbb{P}(noise \mid ctx_j) \cdot \mathbb{P}\Big(entity_j \mid noise\Big) \\ &+ \mathbb{P}(\neg noise \mid ctx_j) \cdot \mathbb{P}\Big(mention_j, entity_j \mid y_k\Big)\end{aligned}$$

where the noise probability for each context, $\mathbb{P}(noise \mid ctx_j)$, is computed using training set.

Finally, for a leaderboard $y_k = (y_k^{(task)}, y_k^{(dataset)}, y_k^{(metric)})$ we assume that $\mathbb{P}(mention, entity \mid y_k) \propto \mathbb{P}\Big(mention \mid y_k^{(entity)}\Big)$, that is, a metric mention "acc" has the same conditional probability for leaderboard (Image Classification, ImageNet, Accuracy) as for (Natural Language Inference, SNLI, Accuracy).

We compute $\mathbb{P}\Big(mention \mid y_k^{(entity)}\Big)$ to be inversly proportional to the number of other entities of type *entity* with the same *mention* evidence (see Appendix D for details).

3.5 Filtering

The final step of AxCell is to filter out results with a linking score that is too low, results for cited models and inferior results (to keep only the best performing results). First, we filter out records not associated with models introduced in a paper being processed. We then remove records for which a linking score is below some given threshold. The remaining records are grouped by leaderboard and for each leaderboard only the best result is kept, based on *higher is better* annotation available in taxonomy; e.g. *Accuracy* would keep higher values, *Error Rate* would keep lower values. Finally, we remove all results with a linking score below the second threshold. This gives us the final list of results tuples extracted from the paper.

4 Dataset

In this section we explain the datasets we used for training and evaluating AxCell for results extraction. The primary input we use for a training dataset is LaTeX source code of machine learning papers from arXiv.org. Over 90% of considered papers have source code available. This allows us to obtain a high quality dataset without common artifacts that arise from extracting data directly from PDF files (Hou et al., 2019).

For training our model we use two main datasets:

- ArxivPapers: An unlabelled dataset of over 100,000 machine learning papers. Used for language model pre-training.
- SegmentedTables: A table segmentation dataset where each cell is annotated according to whether it is a paper, metric, dataset, and so on. Used for table segmentation and table type classification.

We tune the linking and filtering performance of our model using a validation dataset:

- LinkedResults: An annotated dataset of over 200 papers with results tuples, capturing the performance of models in the papers, and links to tables.

Lastly we evaluate the end-to-end performance of our model on our test set:

- PWC Leaderboards: An annotated dataset of over 2,000 leaderboards with results tuples. Used for end-to-end performance evaluation.

We now describe in detail these datasets.

4.1 arXiv Papers

The dataset contains 104,723 papers published on arXiv.org between 2007–2020. 94,616 papers are available with LaTeX sources, from which we extracted 277,996 tables in total. Due to licensing limitations the dataset we release with this paper contains only metadata (available in the public domain) and links to articles. The dataset is unlabeled, designated for use in self-supervised pretraining.

4.2 Segmented Tables

This is a dataset for table classification and segmentation, containing 1400 annotated tables from 354 articles. The dataset provides data on dataset mentions in captions, the type of table (`leaderboard`, `ablation`, `irrelevant`) and ground truth cell annotations into classes: `dataset`, `metric`, `paper model`, `cited model`, `meta` and `task`.

4.3 Linked Results

This is a set of 236 papers we annotated with 1148 results tuples, capturing the performance of models in the papers. Additionally we include metrics scores in a normalized form. We also record metadata such as the names of the models used in papers. Each results tuple (task, dataset, metric name, metric value) is linked to a particular table, row and cell it originates from. Note that results that appear outside of a table, for instance in the paper's text or graphs, are not present in this dataset.

4.4 PWC Leaderboards

This is a dataset of 2,295 leaderboards obtained from the Papers With Code arXiv.org labelling interface. This interface allows an annotator to take a paper and label it with results tuples. It is therefore a good ground-truth test on which to evaluate the end-to-end performance of our automated solution. Additionally, each record is linked to a cell it appears in.

Table 1: End-to-end extraction results on subset of NLP-TDMS (Exp) dataset.

Method	Micro			Macro		
	P	R	F_1	P	R	F_1
(task, dataset, metric)						
TDMS-IE	53.4	66.3	59.2	57.1	66.1	58.5
AxCell	65.8	58.5	61.9	56.0	55.8	54.1
(task, dataset, metric, score)						
TDMS-IE	6.8	8.4	7.5	8.6	9.5	8.8
AxCell	27.4	24.4	25.8	20.2	20.6	19.7

5 Experiments

We now evaluate the end-to-end performance of AxCell on the results extraction task. We evaluate on two datasets: the NLP-TDMS dataset introduced in Hou et al. (2019), in order to compare our method to the state of the art, and on our PWC Leaderboards dataset, which contains many more leaderboards and acts as a more challenging benchmark.

5.1 NLP-TDMS Results

We compare AxCell to the TDMS-IE model from Hou et al. (2019) on the NLP-TDMS dataset in Table 1. The NLP-TDMS (Full) dataset contains 332 papers related to Natural Language Processing with 848 performance annotations of task, dataset, metric and score and 168 unique leaderboards. The subset NLP-TDMS (Exp) is limited to 77 leaderboards appearing in at least 5 papers. See Table 7 in the Appendix for dataset statistics. To compare with Hou et al. (2019), we use the Exp dataset.

Hou et al. (2019) extract records directly from PDF, so the methods are not fully comparable. In order to run AxCell on that dataset we limit the dataset to papers for which LaTeX source code is available. Table 1 shows results on that subset with TDMS-IE performance computed based on published predictions. Our solution yields significantly better results for whole records retrieval despite not being trained on their taxonomy (i.e., the zero-shot scenario in Hou et al. (2019)).

5.2 PWC Leaderboards Results

Having validated the performance of our approach compared to the state of the art, we now apply it to our much larger dataset of leaderboards. Compared to the NLP-TDMS dataset, whose taxonomy consists of 77 leaderboards,

Table 2: Extraction results of AxCell on PWC Leaderboards dataset (restricted to our taxonomy) for entire records (TDMS), records without score (TDM) and individual entities.

Entity	Micro			Macro		
	P	R	F_1	P	R	F_1
TDMS	37.4	23.2	28.7	24.0	21.8	21.1
TDM	67.8	47.8	56.1	47.9	46.4	43.5
Task	70.6	57.3	63.3	60.7	62.6	59.7
Dataset	70.2	48.4	57.3	53.5	52.7	49.9
Metric	68.8	58.5	63.3	58.4	60.4	56.5

our taxonomy consists of 3,445 leaderboards making prediction much more challenging.

The results of our approach for extracting each entity are detailed in Table 2. We achieve reasonable performance on extracting the full TDMS (task, dataset, metric, score) tuple, which is the most challenging setting and the highest scores for extracting task and metric information. The lower scoring entities are generally the ones that depend on the quality of extraction of other entities. For example, extracting leaderboards depends on how well we extract task, dataset and metric entities.

6 Performance Studies

In this section, we perform experiments over the various steps of AxCell in order to better understand their relative importance. Our key finding is that table segmentation is the performance bottleneck of AxCell. We run our experiments on the SegmentedTables dataset introduced in Section 4.2.

6.1 Table Type Classification

The biggest issue of table type classification is in distinguishing between leaderboard and ablation tables (see Figure 5 in Appendix for the confusion matrix). These tables can be very similar structurally: ablations may even compare on the same split of data as the primary result. As the distinction is not always clear, during results retrieval we extract results from both types of tables and pick only the best result during filtering (i.e., the highest or lowest based on predicted metric).

6.2 Table Segmentation

One goal of table segmentation is to generalise to extract tables from unseen tasks. To study this, we partitioned SegmentedTables dataset into 11 folds, based on the task name extracted from paper abstracts. The fold with tables from Image Classification papers is always used as a validation set. For each of the remaining 10 folds we train 5 models with a given fold used as a test set and the other 9 folds used as training data. The final table segmentation model used in AxCell is the one with the highest micro F_1 score on the validation set.

Table 3: Table segmentation results for 10-fold training with image classification papers fixed as a validation set and variable test set. Micro precision, recall and F_1 score are averaged over 5 runs.

	validation			test		
test set	P	R	F_1	P	R	F_1
image gen.	84.5	87.9	86.2	73.4	81.6	77.3
misc.	84.0	88.2	86.0	81.7	93.5	87.2
machine trans.	83.1	90.8	86.8	80.5	94.4	86.9
NLI	83.6	89.6	86.5	84.5	97.3	90.4
object detection	81.9	91.4	86.3	83.7	96.7	89.7
pose estimation	85.1	89.9	87.4	86.0	96.8	91.1
question ans.	83.6	89.5	86.4	80.4	89.6	84.8
semantic seg.	81.4	91.1	86.0	90.2	95.9	92.9
speech rec.	84.7	89.8	87.2	67.2	90.7	77.1
text class.	83.9	90.4	87.0	74.9	93.3	83.1

Table 3 shows micro precision and recall of classifying each non-numeric cell into one of 5 exclusive classes: dataset, metric, competing model, paper's model or other.

We can see that we achieve strong results on all tasks, although some tasks perform better than others. A task like semantic segmentation has less table and benchmark diversity, so benchmark tables for datasets like Cityscapes and PASCAL VOC 2012 are fairly standardised across papers. This makes extraction fairly straightforward. In contrast, the worse performing tasks are unusual in their own way. In image generation, for instance, we are less able to extract the correct dataset entity, whereas in speech recognition, our model has more problems distinguishing paper models from competing models; see Figure 4 in the Appendix.

6.3 Linking

To evaluate linking performance in isolation of other steps we run it on tables with ground truth type and segmentation annotations. The annotations are available in the SegmentedTables dataset for 24 Speech Recognition and 32 Semantic Segmentation papers with 287 annotated leaderboard records in total. For each cell with associated leaderboard annota-

tion we generate cell contexts and use linking to retrieve the top-5 predictions. We test four approaches to generate evidence of mentions.

Bag-of-Words The full name and any word (which is not an English stop-word) occurring in the name of a metric or dataset (as found in taxonomy) is evidence of mention.

Abbreviations We run an abbreviation detector (Neumann et al., 2019) over the ArxivPapers dataset to extract pairs of common abbreviations and their full forms. The previous approach is extended with abbreviations of full forms occurring in name of metric or dataset. For example, with a pair (*en-vi*, *English-Vietnamese*) and dataset name *IWSLT2015 English-Vietnamese*, *en-vi* is added as mention evidence for this dataset.

Manually Curated We extend the Bag-of-Words approach with list of manually curated mention evidence. Only mentions of datasets and metrics related to speech recognition and semantic segmentation are modified.

Combined The previous approach extended with abbreviations.

In Table 4 we show Top-1 and Top-5 accuracy of the predictions over all leaderboard records from each collection of papers. Using abbreviations significantly improves the performance over bag-of-words approach. The worse performance caused by adding abbreviations to manually curated lists suggests that abbreviations could increase rate of false-positive matches of mentions. Another explanation is that manually curated lists of mentions could be biased towards leaderbords related to speech recognition and semantic segmentation due to construction of the lists.

The overall performance of the linking step allows us to use it in production environment for efficient semi-automated extraction of results. Our solution proposes to users the Top-5 predictions associated with cells they pointed, thus eliminating the tedious and error-prone step of matching the results with existing leaderboards and ensuring that metric values are correctly normalized.

Table 4: Linking performance using ground truth annotations of table types and segmentation.

Top-1 Accuracy [%]								
evidence	speech rec.				sem. segmentation			
	TDMS	T	D	M	TDMS	T	D	M
BoW	42	86	45	72	49	95	71	67
abbrs	56	87	57	74	56	95	79	74
curated	76	87	77	87	77	95	89	87
combined	67	87	68	78	72	95	86	85
Top-5 Accuracy [%]								
evidence	speech rec.				sem. segmentation			
	TDMS	T	D	M	TDMS	T	D	M
BoW	72	88	73	84	82	99	89	93
abbrs	76	89	76	84	93	100	94	99
curated	85	90	85	91	97	99	99	99
combined	81	89	81	89	97	99	99	99

7 Future Work

We cover three possible extensions to our work for future research.

First, we might want to consider methods that retrieve ***all*** results rather than just the principal results introduced in the paper. This includes extracting ablation studies to enable search over fine-grained comparison results.

Secondly, we could look more into automatic taxonomy discovery. Currently, we assume a closed-domain approach with taxonomy of leaderboards known in advance. While manually extending the taxonomy requires only adding the task, dataset and metric names, it becomes problematic to cover large fraction of the papers due to publication rate and long tail of leaderboards.

Finally, to relax the necessity of AxCell to have access to LaTeX source we consider using the ArxivPapers dataset as a corpus to train extraction working directly with PDF files.

8 Conclusions

We presented an end-to-end model for extracting results from machine learning papers. Our method performs well across various tasks and leaderboards within machine learning, with a taxonomy that can be easily extended without retraining. Additionally we released a new collection of datasets for training and evaluating on the results extraction task. These datasets enable the training of more fine-grained feature extractors and detailed error analysis. We demonstrated that our approach achieves significant performance gains over the state-of-the-art. Future work may want to build on our

approach for more comprehensive extraction tasks, focussing on more types of result, as well as other information contained in papers such as architectural details and hyperparameters.

Acknowledgements

The authors would like to thank Waleed Ammar, Sebastian Kohlmeier and Iz Beltagy on useful discussion and feedback.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Majid Ghasemi-Gol and Pedro A. Szekely. 2018. Tabvec: Table vectors for classification of web tables. *CoRR*, abs/1802.06290.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of Tasks, Datasets, Evaluation Metrics, and Numeric Scores for Scientific Leaderboards Construction. In *Proceedings of ACL 2019*, pages 5203–5213.

Jeremy Howard and Sylvain Gugger. 2020. fastai: A layered API for deep learning. *Information*, 11.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Nikola Milosevic, Cassie Gregson, Robert Hernandez, and Goran Nenadic. 2019. A framework for information extraction from tables in biomedical literature. *International Journal on Document Analysis and Recognition (IJDAR)*.

Tom Mitchell. 2006. The discipline of machine learning. *Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University*.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing.

David Pinto, Andrew McCallum, Xing Wei, and W. Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, page 235–242, New York, NY, USA. Association for Computing Machinery.

Sebastian Ruder. 2018. Tracking the Progress in Natural Language Processing.

Mayank Singh, Rajdeep Sarkar, Atharva Vyas, Pawan Goyal, Animesh Mukherjee, and Soumen Chakrabarti. 2019. Automated early leaderboard generation from comparative tables. In *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 244–257. Springer.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mingxing Tan and Quoc V. Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*.

Xing Wei, Bruce Croft, and Andrew Mccallum. 2006. Table extraction for answer retrieval. *Inf. Retr.*, 9(5):589–611.

Appendix

A Training Details

Table 5: ULMFiT architecture and hyperparameters used for table type classification and table segmentation.

vocabulary size	30K
tokenization	unigram model
RNN type	LSTM
recurrent layers	3
embeddings dimension	400
hidden state dimension	1152
pretraining	12 epochs
batch size	256

Table 6: Features For Table Segmentation

Feature	Description
is emphasised	whether text in cell is boldfaced, colored, etc.
cell style	e.g. "align-left top-border"
text	mentions of cell's content (as in Figure 3)
cell content	cell's content without styles and references, e.g. "ULMFiT"
row context	concatenated cell's row, e.g. "ULMFiT <sep> 94.5% <sep> 92.1% <sep>"
column context	concatenated cell's column, e.g. "Method <sep> LSTM <sep> GRU <sep> ULMFiT <sep> BERT"
cell reference	list of reference ids used in cell, e.g. "bib4, bib18"

B Datasets

Table 7: Statistics of the NLP-TDMS Full and Exp datasets.

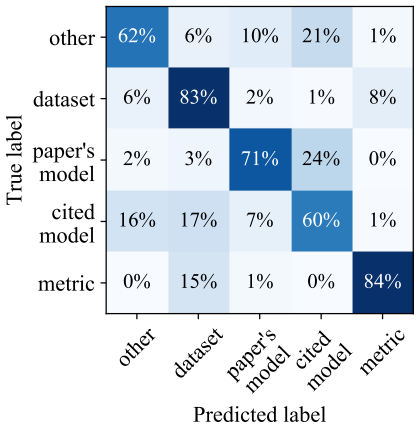
	Full	Exp
unique taxonomy entries	168	77
unique tasks	35	18
unique datasets	99	44
unique metrics	72	30
papers	332	332
results	848	606

Table 8: Statistics for the PWC Leaderboards dataset with all entries (Full) and entries restricted to our taxonomy (Restricted).

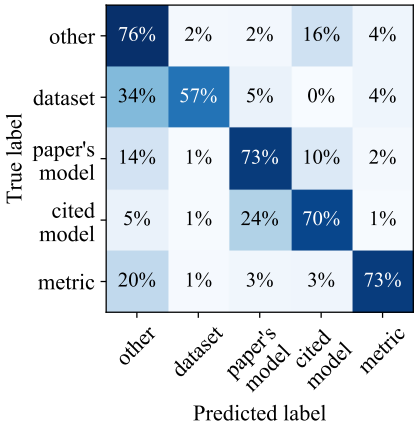
	Full	Restricted
unique taxonomy entries	2295	649
unique tasks	252	134
unique datasets	1156	433
unique metrics	414	162
papers	733	516
results	5406	2802

C Additional Results

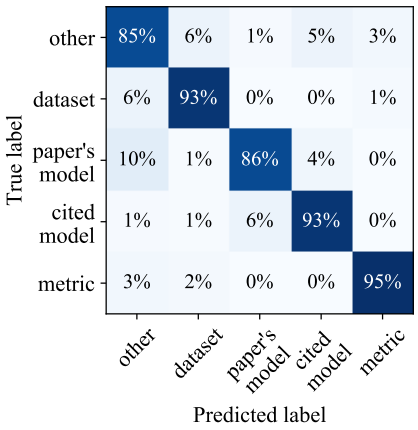
Figure 4: Confusion matrices of segmenting cells into five classes: dataset (including subdatasets), metric, model introduced in processed paper, competing model and other. Results averaged over 5 runs for each task, using 10-fold training as described in Section 6.2.



(a) Speech Recognition

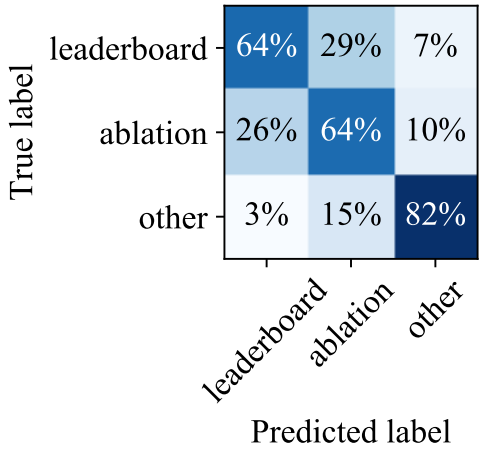


(b) Image Generation



(c) Semantic Segmentation

Figure 5: Confusion matrix of table type classification step.



D Mention Probabilities

Using the methodology from Section 3.4.1, we can calculate $\mathbb{P}(y_k \mid E)$ by combining mentions $\mathbb{P}\left(mention \mid y_k^{(entity)}\right)$. We simplify the notation by referring to this conditional distribution as $\mathbb{P}(m \mid f)$ in this section. This denotes the probability that a mention evidence m for given *entity* f appears in a particular context of cell containing results for leaderboard with entity f.

We compute all possible mentions directly from tasks, datasets and metrics names appearing in leaderboards. For a name of dataset or metric the mentions list consists of the whole name as well as each word, without duplicates and English stop words. As tasks names often consist of common words, to limit the number of false positives the mentions list for a given task contains only that task's name.

We compute probability $\mathbb{P}(m \mid f)$ assuming all mentions (separately for tasks, datasets and metrics) are distributed uniformly, $\mathbb{P}(f_1 \mid m) = \mathbb{P}(f_2 \mid m)$ for all f_1, f_2 for which m is a mention evidence. We then use Bayes rule to get $\mathbb{P}(m \mid f)$, assuming that all mentions of a given type are distributed uniformly. This results in the conditional probability of a mention being inversely proportional to the number of entities having that mention evidence in common:

$$\mathbb{P}(m \mid f) \propto \frac{1}{|\{g : m \text{ is mention evidence for } g\}|}.$$