Dual-Objective Fine-Tuning of BERT for Entity Matching

Ralph Peeters
Data and Web Science Group
University of Mannheim
Mannheim, Germany
ralph@informatik.uni-mannheim.de

Christian Bizer
Data and Web Science Group
University of Mannheim
Mannheim, Germany
chris@informatik.uni-mannheim.de

ABSTRACT

An increasing number of data providers have adopted shared numbering schemes such as GTIN, ISBN, DUNS, or ORCID numbers for identifying entities in the respective domain. This means for data integration that shared identifiers are often available for a subset of the entity descriptions to be integrated while such identifiers are not available for others. The challenge in these settings is to learn a matcher for entity descriptions without identifiers using the entity descriptions containing identifiers as training data. The task can be approached by learning a binary classifier which distinguishes pairs of entity descriptions for the same real-world entity from descriptions of different entities. The task can also be modeled as a multi-class classification problem by learning classifiers for identifying descriptions of individual entities. We present a dual-objective training method for BERT, called JointBERT, which combines binary matching and multi-class classification, forcing the model to predict the entity identifier for each entity description in a training pair in addition to the match/non-match decision. Our evaluation across five entity matching benchmark datasets shows that dual-objective training can increase the matching performance for seen products by 1% to 5% F1 compared to single-objective Transformer-based methods, given that enough training data is available for both objectives. In order to gain a deeper understanding of the strengths and weaknesses of the proposed method, we compare JointBERT to several other BERT-based matching methods as well as baseline systems along a set of specific matching challenges. This evaluation shows that JointBERT, given enough training data for both objectives, outperforms the other methods on tasks involving seen products, while it underperforms for unseen products. Using a combination of LIME explanations and domain-specific word classes, we analyze the matching decisions of the different deep learning models and conclude that BERT-based models are better at focusing on relevant word classes compared to RNN-based models.

PVLDB Reference Format:
Ralph Peeters and Christian Bizer. Dual-Objective Fine-Tuning of BERT for Entity Matching. PVLDB, 14(10): 1913 - 1921, 2021.
doi:10.14778/3467861.3467878

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/wbsg-uni-mannheim/jointbert.

Proceedings of the VLDB Endowment, Vol. 14, No. 10 ISSN 2150-8097.
doi:10.14778/3467861.3467878

1 INTRODUCTION

In practical data integration settings, shared identifiers are often available for a subset of the entity descriptions to be integrated. For instance, some e-shops annotate product offers with GTIN numbers while others do not. The same is true for financial information providers which might identify companies using DUNS numbers, or libraries which might provide ISBN, LCCN, GND or ORCID identifiers. The challenge in these settings is to learn a matcher for entity descriptions without identifier using the entity descriptions having an identifier as training data.

The matching tasks in these settings often involve a set of popular head entities for which a large number of entity descriptions including identifiers is available from different sources. But the tasks also involve long-tail entities for which hardly any data is available. The users of applications that build on the integrated data expect data describing head entities to exhibit hardly any integration errors while errors on tail entities might be more tolerable, e.g. in the context of a price portal or electronic marketplace, users would likely expect data describing a widely-sold phone to be correct, while they would be more forgiving if offers for a tail product are mixed up. This means that matching methods should excel on both, head and tail entities, while taking full advantage of the large amounts of training data that are available for head entities. For head entities, solving the entity matching problem using multi-class classification works well as enough training data is available for learning classifiers that identify descriptions of individual entities, while for long tail entities this is often not the case due to the limited amount of examples. A further drawback of multi-class classification is that entities *unseen* during training cannot be classified and need to be summarized into a class *other*. For long-tail entities and unseen entities, the binary matching approach is often more suited due to its ability to learn matching patterns that generalize.

In this paper, we demonstrate that by combining binary matching and multi-class classification it is possible to better exploit the training data that is available in the multi-source matching scenarios described above and to thus improve the overall matching performance. We present a dual-objective training method for BERT [9], called *JointBERT*, which combines binary matching and multi-class classification. In addition to the binary matching objective, the model is tasked with predicting the entity identifier of each of the two entity descriptions in a pair during training. The idea behind this dual-objective training is to force the model to perceive the matching task not only as a comparison of two isolated entity descriptions but to also incorporate information from other descriptions of the respective entities seen during training into the matching decision. In situations where training data is scarce for

each entity in a pair, the model can still rely on learning patterns using the binary matching objective.

We compare JointBERT to BERT [9], RoBERTa [22], Ditto [20], Deepmatcher [23], Magellan [18], and a baseline using word co-occurrence on five entity matching benchmark datasets. We show that the dual-objective trained JointBERT can improve on single-objective classifiers by 1% to 5% F1 for seen entities, given that enough training data for both objectives is available. We further show that JointBERT performs only slightly worse than BERT when multi-class training data is scarce.

In order to gain a deeper understanding of the strengths and weaknesses of the dual-objective training approach, we evaluate the learned models on specific matching challenges, such as dropped tokens and typos in the entity descriptions as well as entity pairs involving new entities unseen during training. We further investigate which words in the entity descriptions were considered relevant by the different models for specific matching challenges. For this, we generate LIME explanations [28] for sets of matching decisions involving specific challenges and then aggregate the generated LIME word importance weights using domain-specific word classes such as *model name*, *product attribute*, or *stop word*. The comparison of BERT-based models and Deepmatcher shows that the former are better at focusing on strong predictors such as *model numbers* while still being able to fall back to other word classes such as *model name* in cases where *model numbers* are not available.

In summary, the contributions of this paper are:

(1) We present JointBERT, a dual-objective training method for BERT which combines binary entity matching with multi-class classification.
(2) We experimentally compare JointBERT with various other entity matching methods showing that dual-objective training performs best for seen entities, given that enough training data for both objectives is available.
(3) We evaluate the strengths and weaknesses of dual-objective training using specific matching challenges.
(4) We analyze matching decisions of different models by aggregating LIME word weights into domain-specific word classes.

2 RELATED WORK

Entity Matching [1, 5, 6, 14] is the task of identifying entity descriptions that refer to the same real-world entity and has been researched for over 50 years [15]. Entity matching methods can broadly be divided into rule-based, crowd-based, and machine learning-based methods [5, 6, 14]. Since 2018, an increasing number of neural network-based matching methods [13, 23, 30] have been proposed and have pushed the state-of-the-art performance especially for textual entity matching tasks [1]. We include Deepmatcher [23] into our experiments as an example of one of the initial neural network based matching systems.

Transformers for Entity Matching: In natural language processing (NLP), deep Transformer networks [33], pre-trained on large text corpora via language modeling objectives [7, 9, 22] significantly pushed the state-of-the-art in a variety of downstream tasks [16, 34], including a number of sentence-pair classification tasks, e.g. paraphrase identification [11]. Recent studies [3, 20, 31] as well as the dominance of Transformer-based models at the *Semantic Web Challenge on Mining the Web of HTML-embedded Product Data (MWPD)* at ISWC2020 [37] underline the effectiveness of Transformer models like BERT [9] and RoBERTa [22] for the task of entity matching. Recent work [3, 20, 31, 37] on using BERT for entity matching usually builds upon the default BERT architecture for sequence classification. Ditto [20] combines this architecture with injecting domain knowledge as well as data augmentation techniques in pre-processing and fine-tuning.

Multi-Task Learning: Multi-task learning (MTL) [4] has a long history in NLP [29] and generally serves the purpose of complementing a main task using auxiliary training tasks or directly training for multiple main tasks in order to obtain better, more general representations compared to single task training. With the recent successes of neural models, MTL has seen a resurgence, with BERT [9] and other Transformers [19, 36] using a form of MTL for pre-training. Liu et al. [21] recently presented a method to jointly train the lower layers of a BERT model using multiple objectives like sentence classification, sentence similarity, and sentence ranking, resulting in an improved overall performance compared to single task training. JointBERT is inspired by this work and is, up to our knowledge, the first method that applies MTL for entity matching.

3 DUAL-OBJECTIVE TRAINING OF BERT

Problem statement: We address the task of learning a classifier for deciding whether two entity descriptions refer to the same real-world entity or not. Our method assumes that the following two requirements concerning the training set are fulfilled: R1. The training examples consist of pairs of entity description/entity identifier tuples. R2. The training set contains multiple entity descriptions for many of the described entities. These requirements are typically fulfilled in multi-source entity matching settings where a part of the data sources provides entity identifiers, such as GTIN, ISBN, DUNS, or ORCID numbers, together with the entity descriptions.

Model Architecture: BERT is commonly used as follows for entity matching [3, 20, 31, 37]: The string representations of two entity descriptions are concatenated using BERTs *[CLS] Sequence 1 [SEP] Sequence 2 [SEP]* input scheme for sequence classification tasks. The pooled output representation of the [CLS] token from the last encoder layer is then used as input for a linear classification layer followed by sigmoid or softmax to obtain the final class probabilities, in the case of entity matching, *match* and *non-match*. The [CLS] token is used as a representation of the two sequences, trained for a specific task [9]. During BERT pre-training the [CLS] token was tuned for the *Next Sentence Prediction* objective and is adapted to the entity matching objective during fine-tuning for the entity matching task.

JointBERT adds a multi-class training objective to the binary matching objective during the fine-tuning phase. For this objective, the model is tasked to predict the entity identifier of each of the two entity descriptions in a pair in addition to the binary entity matching decision. Figure 1 illustrates the architecture and training objectives of JointBERT. We design the input to the JointBERT model, similar to related work, by (i) concatenating all attributes of an entity description into a single string representation and (ii) combining two entity descriptions into a pair using [CLS] and [SEP]

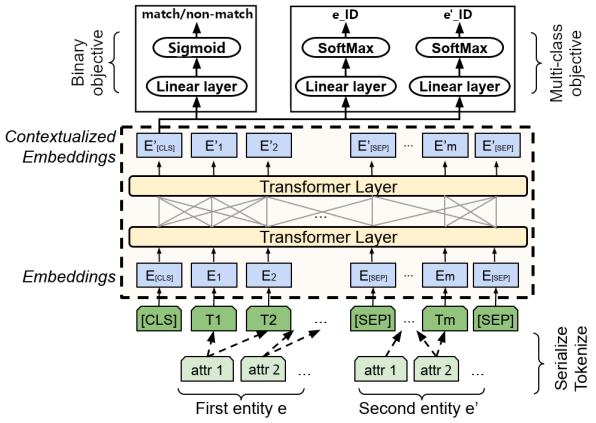

Figure 1: JointBERT architecture. Picture adapted from [20].

tokens as follows: *[CLS] Entity 1 [SEP] Entity 2 [SEP]*. The output representation of the [CLS] token of the last encoder layer is then further fed into three separate linear layers. The first corresponds to the classifier for the binary decision, i.e. do the entity descriptions refer to the same entity or not. The activation function for this layer is the sigmoid function, resulting in the probability of the positive class (match). The second and third linear layers are trained to predict the entity identifier of the left and right entity descriptions respectively. The activation function for both of them is a softmax layer, resulting in the probabilities for each of the entity identifiers in the training set. We use binary cross-entropy loss for the binary objective and cross-entropy loss for both multi-class objectives. With *BCEL* and *CEL* as binary cross entropy and cross entropy loss respectively and $y_{b_i}, y_{l_i}, y_{r_i}$ as binary and multi-class labels, we define the instance loss as:

$$L_i = BCEL\left(y_{b_i}, \hat{y}_{b_i}\right) + \left(CEL\left(y_{l_i}, \hat{y}_{l_i}\right) + CEL\left(y_{r_i}, \hat{y}_{r_i}\right)\right)$$

JointBERT is initialized with the pre-trained BERT_{base} parameters. During training, both objectives are jointly optimized. Matching decisions during inference (application of the trained model) are solely based on the output of the binary classification layer.

4 EXPERIMENTS

We compare the performance of JointBERT to the performance of BERT, RoBERTa, Ditto, Deepmatcher, Magellan, and a word co-occurrence baseline using five entity matching benchmark datasets. Two of these datasets, *WDC LSPC* [27] and *DI2KG monitors* [8], model multi-source settings and fulfil the requirement from the problem statement in Section 3 that multiple entity descriptions should be available for many of the described entities. The other three datasets, *abt-buy*, *dblp-scholar* and *company* [23], do not fulfill this requirement and are included in order to evaluate the performance of JointBERT in traditional two-source settings.

4.1 Datasets

Tables 1 and 2 provide statistics about the training and test sets that we use for the experiments. Below, we describe the datasets.

Table 1: Statistics about the training sets.

Training Set	Size	# Pos. Pairs	# Neg. Pairs	# entities	% entities with min # of descriptions	
					5	10
WDC computers	xlarge	9,690	58,771	745	57	10
	large	6,146	27,213	745	57	10
	medium	1,762	6,332	745	55	8
	small	722	2,112	745	28	0
WDC cameras	xlarge	7,178	35,099	562	50	14
	large	3,843	16,193	562	50	14
	medium	1,108	4,147	562	47	11
	small	486	1,400	562	22	1
WDC watches	xlarge	9,264	52,305	615	62	15
	large	5,163	21,864	615	62	15
	medium	1,418	4,995	615	59	12
	small	580	1,675	615	30	0
WDC shoes	xlarge	4,141	38,288	562	42	2
	large	3,482	19,507	562	42	0
	medium	1,214	4,591	562	39	0
	small	530	1,533	562	13	0
DI2KG monitor	default	611	68,538	100	59	1
abt-buy	default	822	6,837	992	0	0
dblp-scholar	default	4,277	18,688	2,312	12	1
company	default	22,560	67,569	22,560	0	0

Table 2: Statistics about the test sets.

Test Set	# entities w/ pos (overall)	# Pos. Pairs	# Neg. Pairs	# Comb. Pairs
WDC computers	150 (745)	300	800	1,100
WDC cameras	150 (562)	300	800	1,100
WDC watches	150 (615)	300	800	1,100
WDC shoes	150 (562)	300	800	1,100
DI2KG monitor-seen	84 (88)	1,369	122,882	124,251
DI2KG monitor-unseen	123 (141)	7,651	852,365	860,016
DI2KG monitor-combined	207 (229)	9,020	975,247	984,267
abt-buy	205 (819)	206	1,710	1,916
dblp-scholar	848 (1,635)	1,070	4,672	5,742
company	5,640 (5,640)	5,640	16,863	22,503

WDC LSPC datasets: We use the training and test sets from the WDC Product Data Corpus for Large-scale Product Matching (WDC LSPC)[1] [27] for the evaluation. The datasets were built by extracting product offers from the Common Crawl. More specifically, product offers were gathered from e-shops that use schema.org annotations to mark up offer titles, product descriptions, and product IDs like *GTIN* or *MPN* numbers within their HTML pages. WDC LSPC models a multi-source entity matching scenario and fulfills the requirements from Section 3 that entity identifiers as well as multiple entity description should be available for many entities (see last two columns of Table 1). We use the training, validation, and test sets for the four categories *computers*, *cameras*, *shoes* and *watches*. The training sets are available in four sizes, labeled *small*, *medium*, *large* and *xlarge*, ranging from ~2,000 to ~70,000 product offer pairs. All entities that are contained in the test sets are also represented with different entity descriptions in the training set.

[1] http://webdatacommons.org/largescaleproductcorpus/v2/

Thus, WDC LSPC represents the use case of learning a model for *seen* entities. The WDC LSPC datasets were used for the evaluation in the Ditto paper [20] as well as for the *Semantic Web Challenge on Mining the Web of HTML-embedded Product Data* at ISWC2020 [37].

We use the attributes *brand*, *title*, *description* and *specTableContent* for all experiments. The three latter attributes are highly textual and contain longer sequences of words. As the attribute values originate from the Web and may contain noise due to extraction errors, we limit the maximum amount of words in each attribute. This is done with regards to the length limits of BERT-like Transformer models which is usually 512 tokens. We limit the attribute value length to twice their median length in order to ensure this step only affects unusually long strings.

DI2KG monitors dataset: We also evaluate all models using the *DI2KG monitor* dataset [8] that was used for the DI2KG challenge at the DI2KG workshop[2] at VLDB2020. The dataset contains product offers and corresponding ground truth (entity identifiers) from a wide range of online shops and represents a multi-source matching problem. This dataset fits the requirements from Section 3, as entity identifiers are available and multiple entity descriptions exist for most entities (see last two columns of Table 1). We use the pairwise training set offered for the challenge for training. This set was created using a subset of the entity descriptions of a subset of all entities in the ground truth [8]. To allow for unbiased evaluation, we remove all entity descriptions contained in the training set from the collection of entity descriptions in the ground truth before building test sets. This means that the same entity description will not appear in the training and test sets. Using the remaining offers, we build two test sets by creating all possible pairs and assigning the corresponding pair as well as entity identifiers to the two sets (i) entities appearing in the training set (seen) and (ii) entities not appearing the training set (unseen). This allows us to compare the performances of classifiers specifically for these two cases. Each offer in this dataset contains a title attribute as well as a set of specifications which are not aligned across offers originating from different e-shops. We use the title and concatenate all specifications as key-value pairs into a second attribute *specs*. We restrict the length of both attributes in the same way as the length of the attributes in the WDC datasets.

Abt-buy, dblp-scholar, company datasets: We evaluate all models using the *abt-buy*, *dblp-scholar*, and *company* entity matching benchmark datasets. We use the preprocessed versions and splits[3] that were also used for the Deepmatcher paper [23]. The datasets all model the use-case of two mostly deduplicated datasets to be matched for different domains, namely products (abt-buy), scientific texts (dblp-scholar) and companies. We can see in the last two columns of Table 1 that abt-buy and companies do not fulfill the requirement R2 from Section 3 that multiple entity descriptions should be available for a larger fraction of the entities. We include results on these datasets to show how JointBERT performs in such cases. dblp-scholar contains at least 5 entity descriptions for 12% of the entities, likely due to duplicates inside the two sources. We will later see in the results of the experiments that this fraction is too small to fulfill R2. As the datasets only contain pairwise ground truth labels (match/non-match), we use these and the resulting transitive relations from positive pairs to assign entity labels to each entity description in all pairs. We further introduce a summary class *other* and assign to it all entity descriptions which appear only in negative pairs. For each of the three datasets, we use all available attributes during the experiments.

[2]http://di2kg.inf.uniroma3.it/2020/

[3]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

4.2 Models and Baselines

We compare the performance of *JointBERT* to the performance of BERT and RoBERTa. Furthermore, we compare to Ditto [20], the state-of-the-art Transformer-based entity matching framework, the previous state-of-the-art framework Deepmatcher [23], and two non-deep learning baselines. All models are evaluated using precision, recall, and F1 on the positive class (match). The following paragraphs present each of the models as well as the specific settings used for training them.

Transformer-based models: All Transformer-based models are implemented using PyTorch [24] and the HuggingFace Transformers library [35]. We select the uncased base versions of the pretrained models for BERT and RoBERTa for our experiments. For each of them we add a linear layer on top of the pooled output of the [CLS] token and combine it with a sigmoid activation function resulting in a single output probability for the positive class, i.e. match, for the current pair of product offers. All models are trained using binary cross entropy loss. The architecture of JointBERT has already been described in Section 3.

All Transformer-based models are trained on a single NVIDIA RTX 2080Ti GPU with 12GB VRAM. The attributes of each entity description are concatenated into a single string. Any further preprocessing is omitted and left to the tokenizer of the respective models. All models are allowed the full input length of 512 tokens. We fix the batch size at 32 and use the Adam [17] optimizer to train the models for 50 epochs using a linearly decaying learning rate with one epoch warmup. A learning rate sweep is done over the range [1e-5, 3e-5, 5e-5, 8e-5. 1e-4]. Model selection is performed using the maximum F1 value on the validation set. If a models performance on the validation set does not increase over 10 consecutive epochs, training is stopped early. All models are trained three times and we report the average performance.

Ditto: For our experiments with the Ditto [20] framework, we activate the domain knowledge injection module using the offered spans for the product or general domain, depending on the dataset, as well as the training data augmentation module with the operator *span_del*, which deletes randomly sampled spans to augment the data. For the abt-buy, dblp-scholar and company datasets we use the respective operator found working best in the original paper. Instead of using a RoBERTa model at the core of Ditto as in the original paper [20], we use a BERT model in order to make the results more comparable to the results of the JointBERT experiments. The batch size is set to 8 due to memory constraints and we train for 50 epochs using a linearly decaying learning rate of 3e-5 (as in the original paper) with warmup. All Ditto models are trained three times and we report the performance average.

Deepmatcher: For our experiments with Deepmatcher [23], we choose the RNN summarization method which has proven to perform best for the WDC LSPC datasets [26]. We fix the batch size

Table 3: F1 results on the test sets for each dataset.

Testset	Training Set	Word Co-oc	Magellan	Deepmatcher	BERT	RoBERTa	Ditto	JointBERT
WDC computers	xlarge	82.39	63.16	88.95	94.57	94.73	96.53	**97.49**
	large	81.23	64.56	84.32	92.11	94.68	93.81	**96.90**
	medium	70.94	61.59	69.85	89.31	**91.90**	88.97	88.82
	small	62.69	57.60	61.22	80.46	**86.37**	81.52	77.55
WDC cameras	xlarge	73.33	51.70	84.88	91.42	94.39	94.74	**98.02**
	large	76.24	54.49	82.16	91.02	93.91	94.41	**96.51**
	medium	69.89	54.99	69.34	87.02	**90.20**	87.97	87.91
	small	64.86	52.78	59.65	77.47	**85.74**	78.67	78.30
WDC watches	xlarge	79.78	56.04	88.34	95.76	94.87	97.05	**97.09**
	large	79.64	60.59	86.03	95.23	93.93	97.17	**98.46**
	medium	69.54	66.62	67.92	89.00	**92.28**	89.16	87.46
	small	63.49	59.73	54.97	78.73	**87.16**	81.32	75.83
WDC shoes	xlarge	70.38	61.45	86.74	87.44	88.88	93.28	**97.88**
	large	71.18	60.48	83.17	87.37	86.60	90.07	**95.16**
	medium	72.43	59.80	74.40	79.82	81.12	**83.20**	82.61
	small	63.65	58.57	64.71	74.49	**80.29**	75.13	73.13
DI2KG monitor-seen	default	83.59	21.79	77.41	96.22	96.65	86.51	**97.82**
DI2KG monitor-unseen	default	21.24	16.02	33.93	91.49	**93.26**	73.52	82.86
DI2KG monitor-combined	default	33.82	16.90	37.37	92.19	**93.76**	75.28	84.77
abt-buy	default	36.30	43.60	62.80	84.64	**91.05**	82.11	83.44
dblp-scholar	default	85.38	92.30	94.70	95.27	**95.29**	94.47	93.99
company	default	71.81	79.80	**92.70**	91.70	91.81	90.68	91.40

at 16 and set the positive-negative ratio, which controls the class weighting, to the actual distribution of each training set. We keep the default values for all other hyper-parameters. We use fastText embeddings pre-trained on the English Wikipedia[4] as input for Deepmatcher. Each model configuration is trained three times for 50 epochs and we report the averages. For the datasets abt-buy, dblp-scholar and company we report the best result from the original paper [23].

Magellan and Word Co-occurrence: The two non deep learning methods in our experiments are *Magellan* [18], an entity matching framework based on feature similarities, as well as a simple baseline using the binary word co-occurrence among two product offers as features. For the classification step, both methods use scikit-learn [25] classifier implementations.

The *Magellan* framework offers the option of automatically creating similarity features which can then be used for classification. For this purpose, the framework uses a set of human written heuristic rules, which consider for example the average length of text inside an attribute to create meaningful features. We use this automatic feature generation method on the respective training sets to generate feature input for the classifier.

To build the features for the Word Co-occurrence baseline, we first build a vocabulary of words on the respective training set which is subsequently used to build a binary word co-occurrence vector for each entity pair.

We optimize both methods using random search over parameter grids for the following classifiers: Logistic Regression, LinearSVC, DecisionTree, Random Forest and XGBoost. After finding the best hyper-parameter settings as well as classification algorithm on the validation set, the best model configuration is trained three times and we report the averages. For the datasets abt-buy, dblp-scholar and company, we report the result of the Magellan experiments from the Deepmatcher paper [23].

4.3 Results and Discussion

Table 3 shows the F1 results of the experiments across all models and datasets. With regards to the WDC datasets and the *large* and *xlarge* training sizes the JointBERT model performs best and can improve on single-objective BERT by 1% to 10% F1 depending on dataset. The improvement range over RoBERTa and Ditto is generally smaller with a range of 1% to 5% over Ditto apart from the *watches xlarge* training set were both perform equally well. Using the multi-class objective in addition to binary classification results in a strong matching performance of >95% F1 in all categories for the *large* and *xlarge* WDC training sets. For the *medium* and *small* WDC training sets, JointBERT is outperformed by BERT, RoBERTa and Ditto. The performance loss to single-objective BERT is up to 3% F1 and up to 6% for Ditto for these training sizes. This is potentially

[4] https://fasttext.cc/docs/en/pretrained-vectors.html

Table 4: F1 Results for the specific matching challenges of the MWPD test set.

Matching Challenge	# Match	# Non-Match	Word Co-oc	Magellan	Deepmatcher	BERT	RoBERTa	Ditto	JointBERT
Unseen products - high similarity	25	75	48.00	27.89	58.64	**84.53**	83.18	69.97	59.92
Unseen products - low similarity	25	75	11.43	59.04	58.78	76.92	**77.72**	65.78	58.91
Seen products - introduced typos	100	0	54.01	19.79	60.76	71.21	85.49	83.49	**89.08**
Seen products - dropped tokens	100	0	78.79	50.35	71.75	87.62	89.88	90.28	**93.23**
Seen products - very hard cases	25	75	61.22	11.00	74.53	89.04	88.51	94.12	**95.55**
Mix of hard non-matches and matches	250	750	77.93	58.48	79.90	84.08	**86.32**	85.30	84.24
Full MWPD test set	525	975	69.65	48.23	71.53	82.58	**86.20**	83.96	83.35

due to the overall smaller amount of training data in combination with the smaller amount of unique entity descriptions per product which turn the multi-class training objective to having a negative effect on the performance. All BERT-based models consistently beat the baseline methods and Deepmatcher on all training set sizes and categories for the WDC datasets. The differences are most pronounced for smaller training set sizes, making BERT-based models highly training data efficient compared to Deepmatcher, Magellan, and the word co-occurrence baseline.

For the DI2KG monitor dataset RoBERTa is generally slightly ahead of BERT but the best performance for the *seen* test set is achieved by JointBERT which outperforms RoBERTa by nearly 1.5% F1. On the *unseen* test set, BERT and RoBERTa lose around 3-5% performance while JointBERT drops by 15% F1. As the DI2KG monitor dataset provides enough training data for the multi-class objective (see last two columns of Table 1), the impact of this objective on the final prediction is likely high. This seems to lead to very good results when predicting for seen entities and has a negative effect when predicting for unseen entities.

On two of the three two-source datasets *abt-buy*, *dblp-scholar* and *company*, RoBERTa performs best while Ditto and BERT perform slightly worse. On abt-buy, RoBERTa outperforms BERT by 6% F1. The JointBERT results are up to 1% F1 worse than BERT, likely due to the limited amount of distinct entity descriptions per entity in these datasets.

In conclusion, if training data is limited and JointBERT's requirement R2 of multiple entity descriptions for many entities is not fulfilled, then using single-objective training and a robust model like RoBERTa, which has been pre-trained on large amounts of textual data, including web pages, leads to higher performance, especially for unseen entities. If larger amounts of training data for both objectives are available and the task mostly contains seen entities, then JointBERT outperforms RoBERTa.

5 CHALLENGE-SPECIFIC ANALYSIS

In addition to the previous experiments, we further analyze the strong performance of JointBERT when trained using large amounts of training data by evaluating all models, trained with the *computers xlarge* training set[5], on the test set of the *Semantic Web Challenge on Mining the Web of HTML-embedded Product Data* (MWPD) which took place at ISWC2020 [37]. This test set contains 1,500 product offer pairs from the WDC LSPC *computers* category [27]. For the purposes of the challenge, this test set was made intentionally hard by selecting mainly very hard pairs as well as further augmenting some of them to derive subsets of pairs posing specific matching challenges (100 pairs each). These challenges include (i) pairs containing products unseen during training and having a high similarity to seen products, (ii) pairs unseen during training and having a low similarity to seen products, (iii) pairs of seen products which were augmented by manually introducing typos into important words, (iv) pairs of seen products which were augmented by manually dropping important words, and (v) very hard pairs of seen products (i.e. most similar non-matches and least similar matches). Similarity in this regard refers to Jaccard similarity of the title attribute. Apart from these specific matching challenges, the majority of the MWPD test set is comprised of a mix of similar non-matches and similar matches, mainly for seen but also for unseen products.

Table 4 provides statistics about the amounts of matches and non-matches for each challenge and reports the performance of all matchers, trained using the *computers-xlarge* training set, for the specific matching challenges of the MWPD test set. For the two challenges involving unseen products, BERT and RoBERTa perform the best (within 1.5% F1 of each other) with around 84% F1 for unseen products similar to seen products, and around 77% F1 for those which are less similar to seen products. JointBERT's performance for unseen products is the worst among all BERT-based models, which was expected as the multi-class objective steers the model towards recognizing a specific set of entities and reduces its ability to generalize to unseen entities. This is the same effect that was already observed on the monitor-unseen test set in Section 4.

When looking at the challenges involving seen products, which have either typos introduced into important words or in which these words are dropped entirely, JointBERT outperforms all other models by at least 3% F1. The robustness to such challenges likely stems from the additional multi-class objective which supports the binary decision, even though the direct similarity comparison of the two strings gets harder due to typos or dropped words.

The third challenge for seen products consists of especially hard offer pairs. This challenge set is comprised of highly similar negatives as well as dissimilar positives. As for the other two seen product challenges, JointBERT performs best on this challenge and can outperform BERT by 6.5% F1, highlighting the utility of the dual-objective training approach for seen products.

Finally, the last set contains a mix of hard offer pairs for seen products as well as some unseen products and makes up the bulk

[5]Results for other sets are found at https://github.com/wbsg-uni-mannheim/jointbert

of the MWPD test set. All BERT-based models perform comparably here with around 85% F1 apart from RoBERTa which manages to set itself apart slightly with 86.5% F1. Overall, RoBERTa proves to be most robust even though it does not excel over other models at any particular challenge, and performs the best on the mixed set.

6 EXPLAINING MATCHING DECISIONS

In order to better understand the matching decisions of different models, in the following we investigate the importance of words belonging to domain-specific word classes for the matching decisions. We base our analysis on the Mojito [10] framework, which adapts the LIME algorithm [28] for the use case of pairwise entity matching and is part of recent efforts to explain deep entity matching results [12, 32]. LIME creates an explanation for a single matching decision as follows: The instance (pair of entity descriptions) is perturbed using word dropping and labels for all perturbed instances are queried from the model to be explained. This set of instance/label pairs is used afterwards to train a surrogate linear regression model which is presumed to locally approximate the original model. The coefficients of the regression model are extracted and signify the importance of the respective words for the matching decision. Mojito offers a second method for perturbing instances, namely copying tokens between entities, but we decide on using the default LIME word dropping method for our experiments. Figure 2 shows an example of a LIME explanation generated with Mojito for a matching decision by a BERT model for a pair from the MWPD test set. Words with orange coloring signify positive weight (pushing towards match), blue words negative weight (pushing towards non-match). The models decision for non-match seems to be based on the difference in RAM size as well as SSD size between the two Chromebooks, which makes sense for a human.

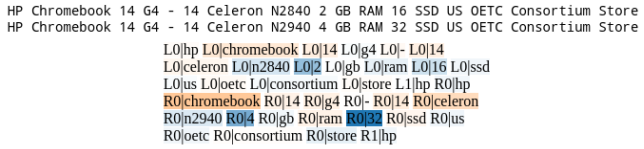

Figure 2: LIME explanation example for a non-match classified correctly by the BERT model.

Aggregating explanations: LIME explanations are restricted to explaining single matching decisions by locally approximating the model around one specific decision. In order to get a better understanding about the types of words onto which a model is focusing in specific matching situations, we want to aggregate the LIME explanations for multiple matching decisions representing a specific matching situation or matching challenge. For aggregating explanations and for abstracting from specific words to more generic word classes relevant for the domain, such as *model number*, *brand* and *model name*, we take a three step approach:

1. Sampling interesting entity pairs: The MWPD test set serves as a basis for building a dataset for explanations, as it contains a variety of hard pairs and products with and without training examples in the associated training set. This allows us to sample interesting entity pairs, whose explanations can give us further insights into the models decision process. As we are interested in the impact of the amount of training data on the matching decisions and we are also interested in the role of strong predictors such as *model numbers* for the matching decisions, we sample 50 pairs, split into 25 positives and 25 negatives, for each of the following classes of entity pairs:

(1) Both products had many training examples (>10)
(2) Both products had few training examples (<5)
(3) Both products had no training examples
(4) Both offers contain a model number
(5) At least one offer does not have a model number

Table 5: Product domain-specific word classes used for manual annotation of the explanation dataset.

Word Class	Examples
Model number	*DUAL-GTX1060-O6G, DT100G3/128GB, ...*
Brand name	*Asus, Apple, Hewlett-Packard, NVIDIA, …*
Model name	*Thinkpad 11e, GTX 1070, Core i7-4770k, ...*
Characteristic attribute	*16 GB RAM, 256 GB SSD, 4 GHz, …*
Stopword	*with, New, Wholesale, Laptop Outlet Direct, …*
Product type	*Hard Disk Drive, Processor, Graphics Card, …*
Other descriptive word	*Kit, OEM, High Performance, …*

2. Annotating entity pairs with word classes: We proceed with manually labeling each of the words in the selected pairs *title* attribute with a domain-specific word class, which we will use later on to aggregate explanation weights and make explanations comparable across different products. Table 5 shows the word classes that we selected as well as examples of words from each class.

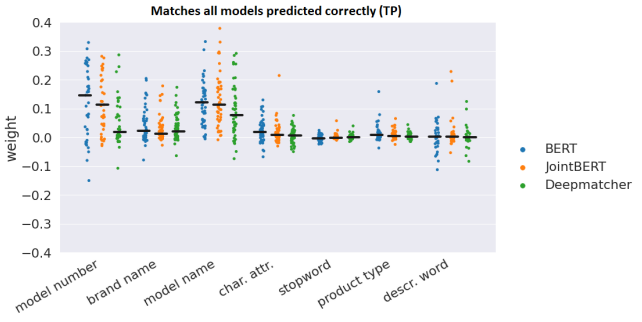

Figure 3: Aggregated explanations for pairs classified correctly by all models.

3. Aggregating explanations for specific subsets: In order to understand the importance of words belonging to domain-specific word classes in specific matching situations and for specific matching challenges, we select subsets of the 250 annotated entity pairs and aggregate the LIME word weights by word classes, and finally compare the resulting weight aggregations between different models. In the following, we analyze two subsets of the annotated pairs: (i) Pairs correctly classified by BERT, JointBERT, and Deepmatcher, (ii) Pairs with and without model numbers.

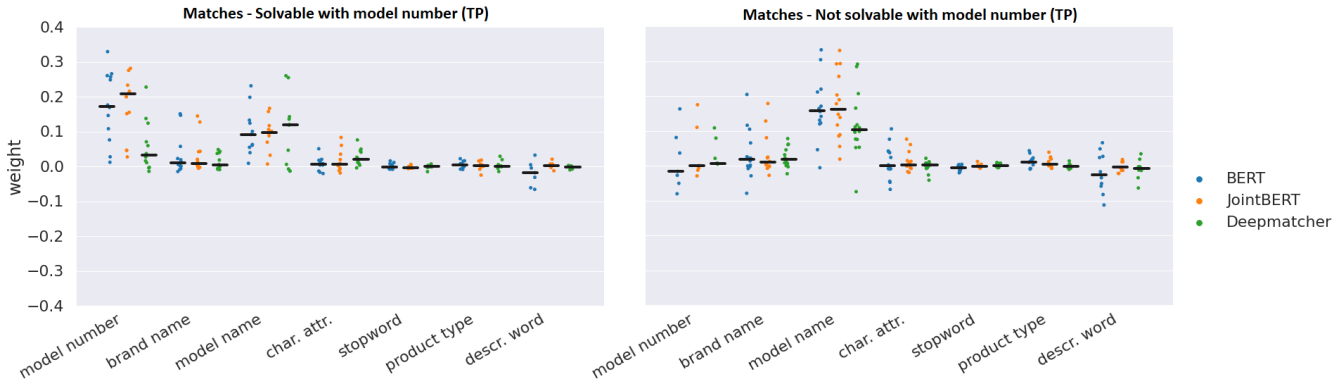

Figure 4: Aggregated explanations for (left) the correctly classified pairs of the group "solvable by model number" and (right) the correctly classified pairs of the group "not solvable by model number".

Correct Matching Decisions: To get a general overview of differences between the three models BERT, JointBERT, and Deepmatcher, we generate the explanation plots using those pairs which all models predicted correctly for the *match* class. We omit *non-matches* due to space reasons - the observed patterns are similar to those for matches and the corresponding plots are available in the repository. Picture 3 shows the resulting visualization. The black lines across the swarm columns for each model denote the median value of that column. There are some notable differences visible between Deepmatcher and the BERT-based models. Both BERT models seem to put a large fraction of the weight on model numbers which is reasonable as this word class is the strongest indicator for matching or non-matching product offers if available. Compared to the BERT models, Deepmatcher puts less weight on model numbers. The second highest weighted word class is model name for the BERT-based models, which is highest weighted for Deepmatcher. Intuitively this makes sense, since the model name is one of the strongest indicators for a matching or non-matching pair as often these are the words where hard non-matching pairs differ in subtle ways (e.g. *IPhone X* vs *IPhone Xs*). For matching pairs these words will likely be exactly the same apart from possible differences due to encoding errors or noise. For the rest of the word classes the differences between the models are minimal. The weighting of stopwords is low for all models, suggesting that the models can mostly ignore irrelevant words.

Role of Model Numbers: We can further drill down using the two subsets *solvable using model numbers* and *not solvable using only model numbers* to illustrate how model decisions change for each of these cases. Figure 4 shows the resulting explanation plots for matches. For the cases which can be solved using model numbers, i.e. they are available for both offers, it is clear that the BERT-based models focus strongly on the model numbers and less on the model name, while the Deepmatcher model is more focused on the model names and does not weight the model number as strongly (left side in Figure 4). When looking at those pairs that are not solvable using model numbers, the BERT-based models shift their focus towards using the model name (right side in Figure 4). The median weight on model name is now even higher than for Deepmatcher. In conclusion, the BERT-based models seem to be able to better focus on strong predictors such as model numbers if available and can better adapt if the strongest predictors, i.e. model numbers, are missing when compared to Deepmatcher, which seems to overly focus on the model name even if model numbers are available.

7 CONCLUSION

We have demonstrated that jointly training BERT for binary and multi-class entity matching outperforms matching models that were trained using only the binary objective by 1% to 5% F1 for seen entities given that sufficient amounts of training data for both objectives are available. This joint training is applicable to any entity matching scenario in which entity identifiers are available for a subset of entity descriptions to be matched. An example of such a scenario are price portals which maintain a product catalog and have to match incoming product offers to it. The analysis of specific matching challenges in Section 5 illustrated that JointBERT outperforms the other models on challenges involving seen products by at least 1.5% F1, again given that enough training data is available. As a downside, JointBERT underperforms on unseen products in comparison to Ditto, BERT and RoBERTa, as the multi-class objective steers the model towards recognizing seen entities. In conclusion, dual-objective training should be considered for use cases involving seen entities and enough labels for both objectives, while single-objective Transformers are more suited for unseen entities and use cases involving small amounts of training data. Aggregating LIME explanations by domain-specific word classes, in an effort to better understand the general focus of the models, uncovered that BERT-based models are more adept in focusing on relevant words such as *model numbers* than Deepmatcher, while still being able to fallback to focusing on other word classes such as *model name* in cases for which *model numbers* are not available. As future work, it would be interesting to investigate combining dual-objective training with data augmentation techniques [20] and additional pre-training using domain-specific corpora [2].

REFERENCES

[1] Nils Barlaug and Jon Atle Gulla. 2021. Neural Networks for Entity Matching: A Survey. *ACM Transactions on Knowledge Discovery from Data* 15, 3 (2021), 52:1–52:37.

[2] Alexander Brinkmann and Christian Bizer. 2021. Improving Hierarchical Product Classification using Domain-specific Language Modelling. In *Proceedings of Workshop on Knowledge Management in e-Commerce*.

[3] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - a Step Forward in Data Integration. In *Proceedings of the International Conference on Extending Database Technology*. 463–473.

[4] Rich Caruana. 1997. Multitask Learning. *Machine Learning* 28 (1997), 41–75.

[5] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer-Verlag, Berlin Heidelberg.

[6] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An Overview of End-to-End Entity Resolution for Big Data. *Comput. Surveys* 53, 6 (2020), 127:1–127:42.

[7] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-Training Text Encoders as Discriminators Rather Than Generators. In *8th International Conference on Learning Representations*.

[8] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, et al. 2021. Alaska: A Flexible Benchmark for Data Integration Tasks. *arXiv:2101.11259 [cs]* (Feb. 2021).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. 4171–4186.

[10] Vincenzo Di Cicco, Donatella Firmani, Nick Koudas, Paolo Merialdo, and Divesh Srivastava. 2019. Interpreting Deep Learning Models for Entity Resolution: An Experience Report Using LIME. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 8:1–8:4.

[11] William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*. 9–16.

[12] Amr Ebaid, Saravanan Thirumuruganathan, Walid G. Aref, Ahmed Elmagarmid, and Mourad Ouzzani. 2019. EXPLAINER: Entity Resolution Explanations. In *2019 IEEE 35th International Conference on Data Engineering*. 2000–2003.

[13] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.

[14] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16.

[15] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[16] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, et al. 2020. XTREME: A Massively Multilingual Multi-Task Benchmark for Evaluating Cross-Lingual Generalisation. In *International Conference on Machine Learning*. 4411–4421.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

[18] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, et al. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1197–1208.

[19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, et al. 2020. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. In *8th International Conference on Learning Representations*.

[20] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[21] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4487–4496.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]* (July 2019).

[23] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, et al. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of machine learning research* 12 (2011), 2825–2830.

[26] Ralph Peeters, Anna Primpeli, Benedikt Wichtlhuber, and Christian Bizer. 2020. Using Schema.Org Annotations for Training and Maintaining Product Matchers. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*. 195–204.

[27] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Companion Proceedings of The 2019 World Wide Web Conference*. 381–386.

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1135–1144.

[29] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv:1706.05098 [cs, stat]* (June 2017).

[30] Kashif Shah, Selcuk Kopru, and Jean David Ruvini. 2018. Neural Network Based Extreme Classification and Similarity Models for Product Matching. In *Proceedings of the 2018 Conference of the Association for Computational Linguistics, Volume 3*. 8–15.

[31] Kai-Sheng Teong, Lay-Ki Soon, and Tin Tin Su. 2020. Schema-Agnostic Entity Matching Using Pre-Trained Language Models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2241–2244.

[32] Saravanan Thirumuruganathan, Mourad Ouzzani, and Nan Tang. 2019. Explaining Entity Resolution Predictions: Where Are We and What Needs to Be Done?. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 1–6.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[34] Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, et al. 2019. Can You Tell Me How to Get Past Sesame Street? Sentence-Level Pretraining Beyond Language Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4465–4476.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, et al. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 38–45.

[36] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8413–8426.

[37] Ziqi Zhang, Christian Bizer, Ralph Peeters, and Anna Primpeli. 2020. MWPD2020: Semantic Web Challenge on Mining the Web of HTML-Embedded Product Data. In *CEUR Workshop Proceedings*, Vol. 2720. 2–18.