Transfer and Multi-Task Learning for Noun–Noun Compound Interpretation

Murhaf Fares **Stephan Oepen** **Erik Velldal**

Department of Informatics
University of Oslo
murhaff|oe|erikve@ifi.uio.no

Abstract

In this paper, we empirically evaluate the utility of transfer and multi-task learning on a challenging semantic classification task: semantic interpretation of noun–noun compounds. Through a comprehensive series of experiments and in-depth error analysis, we show that transfer learning via parameter initialization and multi-task learning via parameter sharing can help a neural classification model generalize over a highly skewed distribution of relations. Further, we demonstrate how dual annotation with two distinct sets of relations over the same set of compounds can be exploited to improve the overall accuracy of a neural classifier and its F_1 scores on the less frequent, but more difficult relations.

1 Introduction

Noun–noun compound interpretation is the task of assigning semantic relations to pairs of nouns (or more generally, pairs of noun phrases in the case of multi-word compounds). For example, given the nominal compound *street protest*, the task of compound interpretation is to predict the semantic relation holding between *street* and *protest* (a locative relation in this example). Given the frequency of noun–noun compounds in natural language – e.g. 3% of the tokens in the British National Corpus (Burnard, 2000) are part of noun–noun compounds (Ó Séaghdha, 2008) – and its relevance to other natural language processing (NLP) tasks such as question answering and information retrieval (Nakov, 2008), noun–noun compound interpretation has been the focus of much work, in theoretical linguistics (Li, 1972; Downing, 1977; Levi, 1978; Finin, 1980; Ryder, 1994), psycholinguistics (Gagné and Shoben, 1997; Marelli et al., 2017), and computational linguistics (Lauer, 1995; Nakov, 2007; Ó Séaghdha and Copestake, 2009; Girju et al., 2009; Kim and Baldwin, 2013; Dima and Hinrichs, 2015). In computational linguistics, noun–noun compound interpretation is, by and large, approached as an automatic classification problem. Hence several machine learning (ML) algorithms and models have been used to learn the semantics of nominal compounds, including Maximum Entropy (Tratz and Hovy, 2010), Support Vector Machines (Ó Séaghdha and Copestake, 2013) and Neural Networks (Dima and Hinrichs, 2015; Vered and Waterson, 2018). These models use information from lexical semantics such as WordNet-based features and distributional semantics such as word embeddings. Nonetheless, noun–noun compound interpretation remains one of the more difficult NLP problems because: 1) noun–noun compounding, as a linguistic construction, is very productive and 2) the semantics of noun–noun compounds is not easily derivable from the compounds' constituents (Vered and Waterson, 2018). Our work, in part, contributes to advancing NLP research on noun–noun compound interpretation through the use of transfer and multi-task learning.

The interest in using transfer learning (TL) and multi-task learning (MTL) in NLP has surged over the past few years, showing 'mixed' results depending on the so-called main and auxiliary tasks involved, model architectures and datasets, among other things (Collobert and Weston, 2008; Mou et al., 2016; Søgaard and Goldberg, 2016; Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017). These 'mixed' results, coupled with the fact that neither TL nor MTL has been applied to noun–noun compounds interpretation before, motivate our extensive empirical study on the use of TL and MTL for compound interpretation, not only to supplement existing research on the utility of TL and MTL for semantic NLP tasks in general, but also to determine their benefits for compound interpretation in particular.

One of the primary motivations for using multi-task learning is to improve generalization by "leveraging the domain-specific information contained in the training signals of *related* tasks" Caruana (1997). In this work, we show that TL and MTL can indeed be used as a kind of *regularizer* to learn to predict infrequent relations given a highly skewed distribution of relations from the noun–noun compound dataset of Fares (2016) which is especially well suited for TL and MTL experimentation as detailed in Section 3.

Our contributions can be summarized as:

1. Through careful result analysis, we find that TL and MTL (mainly on the embedding layer) do improve the overall accuracy and the F_1 scores of the less frequent relations in a highly skewed dataset, in comparison to a strong single-task learning baseline.

2. Even though our work focuses on TL and MTL, to the best of our knowledge, we are the first to report experimental results on the comparatively recent dataset of Fares (2016).

2 Related Work

Noun–Noun Compound Interpretation Existing approaches to noun–noun compound interpretation vary depending on the taxonomy of compound relations as well as the machine learning models and features used to learn those relations. For example, Ó Séaghdha (2007) defines a coarse-grained set of relations (viz. six relations based on theoretical work by Levi (1978)), whereas Tratz and Hovy (2010) assume a considerably more fine-grained taxonomy of 43 relations. Others question the very assumption that noun–noun compounds are interpretable using a finite, predefined set of relations (Downing, 1977; Finin, 1980) and propose alternative paraphrasing-based approaches (Nakov, 2007; Shwartz and Dagan, 2018). We here focus on the approaches that cast the interpretation problem as a classification task over a finite predefined set of relations. A wide variety of machine learning models have been already applied to learn this task, including nearest neighbor classifiers using semantic similarity based on lexical resources (Kim and Baldwin, 2005), kernel-based methods like SVMs using lexical and relational features (Ó Séaghdha and Copestake, 2009), Maximum Entropy models with a relatively large selection of lexical and surface form features such as synonyms and affixes (Tratz and Hovy, 2010) and, most recently, neural networks either solely relying on word embeddings to represent noun–noun compounds (Dima and Hinrichs, 2015) or word embeddings and so-called path embeddings (which encode information about lemmas and part-of-speech tags, inter alia) in a combined paraphrasing and classification approach (Vered and Waterson, 2018). Of the aforementioned studies, Tratz and Hovy (2010); Dima and Hinrichs (2015); Vered and Waterson (2018) have all used the same dataset by Tratz and Hovy (2010). To the best of our knowledge, TL and MTL have never been applied to compound interpretation before, and in the following we therefore review some of the previous work on TL and MTL on other NLP tasks.

Transfer and Multi-Task Learning A number of recent studies have presented comprehensive experiments on the use of TL and MTL for a variety of NLP tasks including named entity recognition and semantic labeling (Martínez Alonso and Plank, 2017), sentence-level sentiment classification (Mou et al., 2016), super-tagging and chunking (Bingel and Søgaard, 2017) and semantic dependency parsing (Peng et al., 2017). The common thread among the findings of these studies is that the benefits of TL and MTL largely depend on the properties of the tasks at hand, such as the skewedness of the data distribution (Martínez Alonso and Plank, 2017), the semantic similarity between the source and target tasks (Mou et al., 2016), the learning pattern of the auxiliary and main tasks where "target tasks that quickly plateau" benefit most from "non-plateauing auxiliary tasks" (Bingel and Søgaard, 2017) and the "structural similarity" between the tasks (Peng et al., 2017). In addition to the difference in the NLP tasks they experiment with, the aforementioned studies assume slightly different definitions of TL and MTL (cf. Section 4). Our work is similar in spirit to that of Peng et al. (2017) in the sense that we use TL and MTL to learn different 'formalisms' (semantic annotations of noun–noun compounds in our case) on the *same* dataset. However, our experimental setup is more similar to the work by Mou et al. (2016) in that we experiment with parameter initialization on all the layers of the neural model and simultaneously train one MTL model on two sets of relations (cf. Section 5).

3 Task Definition and Dataset

Given a set of labeled pairs of nouns, each a noun–noun compound, the task is simply to learn to classify the semantic relations holding between each pair of compound constituents. The difficulty of this task, obviously, depends on the label set used and its distribution, among other things. For all the experiments presented in this paper, we adapt the noun–noun compounds dataset created by Fares (2016) which consists of compounds annotated with two different taxonomies of relations; in other words, for each noun–noun compound there are two distinct relations, drawing on different linguistic schools. The dataset was derived from existing linguistic resources, such as `NomBank` (Meyers et al., 2004) and the Prague Czech-English Dependency Treebank 2.0 (Hajič et al., 2012, `PCEDT`). Our motivation for using this dataset is twofold: first, dual annotation with relations over the same underlying set of compounds maximally enables TL and MTL perspectives; second, alignment of two distinct annotation frameworks over the same data facilitates contrastive analysis and comparison across frameworks.

More specifically, we use a subset of the dataset created by Fares (2016), by focusing on type-based instances of so-called two-word compounds.[1] The original dataset by Fares (2016) also includes multi-word compounds (i.e. compounds consisting of more than two nouns) and more than just one instance per compound type. Furthermore, we define a three-way split of the dataset; Table 1 presents the number of compound types per split and the vocabulary size of each split (i.e. the number of unique words in each split); the latter is also broken down in terms of words occurring in the right-most position (right constituents) and the left-most position (left constituents).[2] Overall, the two label sets consists of 35 so-called `PCEDT` functors and 18 `NomBank` argument and adjunct relations. As detailed in Section 7.1, these label sets are far from being uniformly distributed.

Abstractly, many relations in `PCEDT` and `NomBank` describe similar semantic concepts, since they annotate the semantics of the same text. For example, Fares (2016) reports that the temporal and locative relations in `NomBank` (`ARGM-TMP` and `ARGM-LOC`, respectively) and their counterparts in `PCEDT` (`TWHEN` and `LOC`) exhibit a relatively consistent behavior across frameworks as they annotate many of the same compounds. However, Fares (2016) also points out that some abstractly similar relations do not align well in practice; for example, the functor `AIM` in `PCEDT` and the modifier argument `ARGM-PNC` in `NomBank` express a somewhat similar semantic concept (purpose) but the overlap between the sets of compounds they annotate in practice is rather small. Nonetheless, it is plausible to assume that the semantic similarity in the label sets—whenever it exists—can be exploited in the form of transfer and multi-task learning, not least because the overall distribution of the relations in the two frameworks is different.

[1]Two-word compounds consist of two whitespace-separated constituents. A single constituent, however, can be a 'simple' noun (e.g. *system*) or a hyphenated noun (e.g. *land-ownership*) leading to compounds like *land-ownership system*. The representation of compounds with hyphenated constituents is explained in Section 5.1

[2]We use the terms left and right constituents, instead of modifier and head nouns, because the dataset does not make explicit the notion of 'headedness'.

	Train	Dev	Test
Compounds	6932	920	1759
Vocab size	4102	1163	1772
Right constituents	2304	624	969
Left constituents	2405	618	985

Table 1: Characteristics of the noun–noun compound dataset used in our experiments. The numbers in this table correspond to a (sub)set of the dataset by Fares (2016), see Section 3.

4 Transfer vs. Multi-Task Learning

In this section, we use the notations and definitions by Pan and Yang (2010) to define our setup for transfer and multi-task learning.

Our classification task $\mathcal{T}$ can be defined in terms of all training pairs (X, Y) and a probability distribution $P(X)$, where $X = x_i, \ldots, x_N \in \mathcal{X}$ and $Y = y_i, \ldots, y_N \in \mathcal{Y}$; $\mathcal{X}$ is the input feature space, $\mathcal{Y}$ is the set of all labels and N is the size of the training data. A task's domain $\mathcal{D}$ is defined by $\{\mathcal{X}, P(X)\}$. Our goal is to learn a function $f(X)$ that predicts Y based on the input features X. Assuming two ML tasks, $\mathcal{T}_a$ and $\mathcal{T}_b$, we would train two models (i.e. learn two separate functions f_a and f_b) to predict Y_a and Y_b in a single-task learning setup. However, if $\mathcal{T}_a$ and $\mathcal{T}_b$ are related somehow, either explicitly or implicitly, TL and MTL can improve the generalization of either task or both (Caruana, 1997; Pan and Yang, 2010; Mou et al., 2016). Two tasks are considered related when their domains, $\mathcal{D}_a$ and $\mathcal{D}_b$, are similar but their label sets are different $\mathcal{Y}_a \neq \mathcal{Y}_b$ or when their domains are different but their label sets are identical, i.e. $\mathcal{Y}_a = \mathcal{Y}_b$ (Pan and Yang, 2010).[3] As such, noun–noun compound interpretation over the dataset of Fares (2016) is a well suited candidate for TL and MTL, because the training examples are identical, i.e. $X_{PCEDT} = X_{NomBank}$, but the label sets are different $\mathcal{Y}_{PCEDT} \neq \mathcal{Y}_{NomBank}$.

For the sake of clarity, we distinguish between transfer learning and multi-task learning in this paper, even though these two terms are at times used somewhat interchangeably in the literature. For example, what we call TL and MTL in this paper are both referred to as transfer learning by Mou et al. (2016). We define TL as using the parameters (i.e. weights in neural networks) of one model trained on $\mathcal{T}_a$ to initialize another model for $\mathcal{T}_b$. Mou et al. (2016) refer to this method as "parameter initialization".[4] MTL, on the other hand, here refers to training (parts of) the same model to learn $\mathcal{T}_a$ and $\mathcal{T}_b$, i.e. learning one set of parameters for both tasks. The idea is to train a single model simultaneously on the two tasks where one task is considered to introduce inductive bias which would help the model generalize over the main task. Note, however, that this does not necessarily mean that we eventually want to use a single model to predict both label sets in practice (cf. Section 5.3).

5 Neural Classification Models

Here we present the neural classification models used in our experiments. To isolate the effect of TL and MTL, we first present a single-task learning model, which serves as our baseline model, and then we use the same model to apply TL and MTL.

5.1 Single-Task Learning Model

In our single-task learning (STL) setup, we train and fine-tune a feed-forward neural network based on the neural classifier proposed by Dima and Hinrichs (2015), which consists of: 1) input layer, 2) embedding layer, 3) hidden layer, and 4) output layer. The input layer is simply two integers specifying the indices of a compound's constituents in the embedding layer where the word embedding vectors are stored; the selected word embedding vectors are then fed to a fully connected hidden layer whose size is the same as the number of dimensions of the word embedding vectors. Finally, a *softmax* function is applied on the output layer and the most likely relation is selected.

The compound's constituents are represented using a 300-dimensional word embedding model trained on an English Wikipedia dump (dated February 2017) and English Gigaword Fifth Edition (Parker et al., 2011) using GloVe (Pennington et al., 2014). The embedding model was trained by Fares et al. (2017) who provide more details on the hyperparameters used to train the embedding model.[5] When looking up a word in the embedding model, if the word is not found we check if the word is uppercased and look up the same word in lowercase. If a word is hyphenated and is not found in the embedding vocabulary, we split it on the hyphen and average the vectors of its parts (if they exist in the vocabulary). If after these steps the word is still not found, we use a designated vector for unknown words.

Architecture and Hyperparameters Our choice of hyperparameters is motivated by several rounds of experimentation on the single-task learning model as well as the choices made by Dima and Hinrichs (2015).

The weights of the embedding layer (i.e. the word embeddings) are updated during training in all the models. The optimization function we use in all the models is Adaptive Moment Estimation, known as *Adam* (Kingma and Ba, 2015) with $\eta = 0.001$ (the default learning rate). The loss function is negative-log likelihood (aka categorical cross-entropy). We use a *Sigmoid* activation function on the hidden layer units. All the models are trained using mini-batches of size five. The maximum number of epochs is set to 50, but we also use an early stopping criterion on the model's accuracy on the validation split (i.e. training is interrupted if the validation accuracy doesn't improve over five consecutive epochs). We implement all the models in *Keras* with *TensforFlow* as backend. All the TL and MTL models are trained with the same hyperparameters of the STL model.[6]

[3]When the label sets are identical, TL practically becomes a technique for *domain adaptation*. Though these two terms have also been used interchangeably (Chung et al., 2018).

[4]Using *pretrained* word embeddings as input representation is in a sense a form of unsupervised transfer learning, but in this work we focus on transfer learning based on supervised learning.

[5]vectors.nlpl.eu/repository

[6]github.com/ltgoslo/fun-nom

5.2 Transfer Learning Models

Transfer learning in our experiments amounts to training an STL model on PCEDT relations, for example, and then using (some of) its weights to initialize another model for NomBank relations. Given the architecture of the neural classifier described in Section 5.1, we identify three ways to implement TL: 1) **TL$_E$**: Transfer of the embedding layer weights, 2) **TL$_H$**: Transfer of the hidden layer weights, and 3) **TL$_{EH}$**: Transfer of both the embedding and hidden layer weights. Furthermore, we distinguish between transfer learning from PCEDT to NomBank and vice versa; that is, either task can be used as main task or auxiliary task. Hence, we either start by training on NomBank and use the weights of the corresponding transfer layer to initialize the PCEDT model or the other way around. In total, this leads to six setups, as shown in Table 2. Note that we do not apply TL (or MTL) on the output layer because it is task- or dataset-specific (Mou et al., 2016).

5.3 Multi-Task Learning Models

In MTL, we train one model simultaneously to learn both PCEDT and NomBank relations, and therefore all the MTL models have two objective functions and two output layers. We implement two MTL setups: **MTL$_E$**, which consists of a shared embedding layer but two task-specific hidden layers, and **MTL$_F$**, which, apart from the output layer, does not have task-specific layers, i.e. both the embedding and hidden layers are shared. We distinguish between the auxiliary and main tasks based on which validation accuracy (NomBank's or PCEDT's) is monitored by the early stopping criterion. Hence we end up with a total of four MTL models as shown in Table 3.

6 Experimental Results

Tables 2 and 3 present the accuracies of the different TL and MTL models on the development and test splits in NomBank and PCEDT. The top row in both tables shows the accuracy of the STL model. All the models were trained on the training split only. There are several observations one can draw from these tables. First, the accuracy of the STL models drops when the models are evaluated on the test split, whether on NomBank or PCEDT. Second, all the TL models achieve better accuracy on the test split of NomBank even though transfer learning does not remarkably improve accuracy on the development split of the same dataset. The MTL models, especially MTL$_F$, have a negative effect on the development accuracy of NomBank, but we still see the same improvement, as in TL, on the test split. Third, both the TL and MTL models exhibit less consistent effects on PCEDT (on both the development and test splits) compared to NomBank; for example, all the TL models lead to about 1.25 points absolute improvement in accuracy on NomBank, whereas in PCEDT TL$_E$ is clearly better than the other two TL models (TL$_E$ improves over the STL accuracy by 1.37 points).

Model	NomBank		PCEDT	
	Dev	Test	Dev	Test
STL	78.15	76.75	58.80	56.05
TL$_E$	78.37	**78.05**	59.57	**57.42**
TL$_H$	78.15	78.00	59.24	56.51
TL$_{EH}$	**78.48**	78.00	**59.89**	56.68

Table 2: Accuracy (%) of the transfer learning models.

Model	NomBank		PCEDT	
	Dev	Test	Dev	Test
STL	**78.15**	76.75	58.80	56.05
MTL$_E$	77.93	78.45	**59.89**	**56.96**
MTL$_F$	76.74	**78.51**	58.91	56.00

Table 3: Accuracy (%) of the MTL models.

Overall, the accuracy of the STL models drops when evaluated on the test split of NomBank and PCEDT (in comparison to their accuracy on the development split); this might be an indicator of overfitting, especially because we select the model that performs best on the development split in our stopping criterion. Both TL and MTL, on the other hand, improve accuracy on the test splits, even though the same stopping criterion was used for STL, TL and MLT. We interpret this result as improvement in the models' generalization ability. However, given that these improvements are relatively small, we next take a closer look at the results to understand if and how TL and MTL help.

7 Results Analysis

This section presents a systematic analysis of the performance of the models based on insights from the dataset used in the experiments as well as the classification errors of the models. The discussion in the following sections is based on the results on the test split rather than the development split, primarily because the former is larger in size.[7]

[7]One can also argue that result analysis on the test split is stricter than on the validation split. While using an early stop-

	A0	A1	A2	A3	LOC	MNR	TMP
Count	132	1282	153	75	25	25	27
STL	49.82	87.54	**45.78**	60.81	28.57	29.41	**66.67**
TL_E	**55.02**	87.98	41.61	60.14	27.91	33.33	63.83
TL_H	54.81	87.93	42.51	60.00	25.00	**35.29**	65.31
TL_{EH}	53.62	87.95	42.70	61.11	29.27	33.33	65.22
MTL_E	54.07	88.34	42.86	61.97	**30.00**	28.57	**66.67**
MTL_F	53.09	**88.41**	38.14	**62.69**	00.00	00.00	52.17

Table 4: Per-label F_1 score on the NomBank test split.

	ACT	TWHEN	APP	PAT	REG	RSTR
Count	89	14	118	326	216	900
STL	43.90	42.11	22.78	42.83	20.51	68.81
TL_E	49.37	**70.97**	27.67	41.60	**30.77**	**69.67**
TL_H	53.99	62.07	25.00	**43.01**	26.09	68.99
TL_{EH}	49.08	64.52	**28.57**	42.91	28.57	69.08
MTL_E	**54.09**	66.67	24.05	42.03	27.21	69.31
MTL_F	47.80	42.11	25.64	40.73	19.22	68.89

Table 5: Per-label F_1 score on the PCEDT test split.

7.1 Relation Distribution

To demonstrate the difficulty of the problem at hand, we plot the distribution of the most frequent relations in NomBank and PCEDT across the three data splits in Figure 1. We find that almost 71.18% of the relations in the NomBank training split are of type ARG1 (proto-typical patient), and 52.20% of the PCEDT relations are of type RSTR (an underspecified adnominal modifier). Such highly skewed distributions of the relations makes learning some of the other relations more difficult, if not impossible in some cases. In fact, of the 15 NomBank relations observed in the test split, five relations are never predicted by any of the STL, TL and MTL models, and of the 26 PCEDT relations observed in the test split only six are predicted. That said, the non-predicted relations are extremely infrequent in the training split (e.g. 23 PCEDT functors occur less than 20 times in the training split), and it is therefore questionable if an ML model will be able to learn them under any circumstances.

From this imbalanced distribution of relations, it immediately follows that accuracy alone, as evaluation measure, is not sufficient to identify the best performing model. Therefore, in the following section we report, and analyze, the F_1 scores of the predicted NomBank and PCEDT relations across all the STL, TL and MTL models.

7.2 Per-Relation F_1 Scores

Tables 4 and 5 show the per-relation F_1 scores for NomBank and PCEDT, respectively. Note that we only include the results for the relations that are actually predicted by at least one of the models.

We observe several interesting patterns in Tables 4 and 5. First, the MTL_F model seems to be confusing for both datasets: it leads to substantially degraded F_1 scores on four NomBank relations, including the locative modifier ARGM-LOC and manner modifier ARGM-MNR (shortened to LOC and MNR in Table 4) which the model is no longer able to predict. The same model has the worst F_1 score, compared to all other models, for two PCEDT relations, REG (which expresses a circumstance) and PAT (patient). Given that the MTL_F model achieves the highest accuracy on the NomBank test split (cf. Table 3), it becomes all the more evident that mere accuracy scores are not enough to judge the utility of TL and MTL for this task (and dataset).

Second, with the exception of the MTL_F model, all the TL and MTL models consistently improve the F_1 score of all the PCEDT relations except PAT. Most notably, the F_1 scores of the relations TWHEN and ACT see a remarkable boost, compared to other PCEDT relations, when only the embedding layer's weights are shared (MTL_E) or transfered (TL_E). This result can be partly explained by looking at the correspondence matrices between NomBank arguments and PCEDT functors shown in Tables 7 and 6, which show how the PCEDT functors map to NomBank arguments in the training split (Table 6) and the other way around (Table 7). From Table 6, we see that 80% of the compounds annotated as TWHEN in PCEDT were annotated as ARGM-TMP in NomBank. In addition, 47% of ACT (Actor) relations map to ARG0 (Proto-Agent) in NomBank, even though this mapping is not as clear as one would have hoped, it is still relatively high if we consider how other PCEDT relations map to ARG0. The correspondence matrices also show that the assumed theoretical similarities between the NomBank and PCEDT relations do not always hold. Nonetheless, even such 'imperfect' correspondence can provide a 'training signal' that help the TL and MTL models learn relations such as TWHEN and ACT.

Since the TL_E model outperforms STL on predicting REG by ten absolute points, we inspected

ping criterion based on the validation data can help prevent overfitting on the training data, we still choose a model that achieves the best accuracy on the validation split. In addition, it's unclear if early stopping helps when the validation split is not fully representative of the problem (Prechelt, 2012).

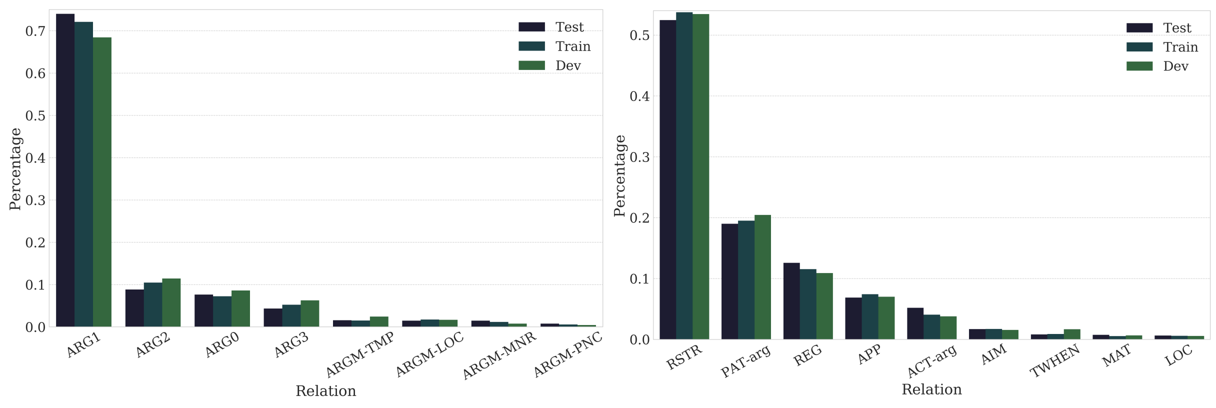

Figure 1: Distribution of `NomBank` relations (left) and `PCEDT` relations (right)

all the `REG` compounds that were correctly classified by the TL$_E$ model but were misclassified by the STL model and found that the latter misclassified them as `RSTR` which indicates that TL from `NomBank` helps the TL$_E$ model recover from the STL's over-generalization in `RSTR` prediction.

The two `NomBank` relations that receive the highest boost in F$_1$ score (about five absolute points) are `ARG0` and `ARGM-MNR`, but the improvement in the latter relation corresponds to only one more compound which might well be predicted correctly by chance. Overall, TL and MTL from `NomBank` to `PCEDT` is more helpful than the other way around. One way to explain this is considering the first rows in Tables 6 and 7, where we see that five `PCEDT` relations (including the four most frequent ones) map to `ARG1` in `NomBank` in more than 60% of the cases for each relation. This means that the weights learned to predict `PCEDT` relations offer little or no inductive bias for `NomBank` relations. Whereas if we consider the mapping from `NomBank` to `PCEDT`, we see that even though many `NomBank` arguments map to `RSTR` in `PCEDT` the percentages are lower, and hence the mapping is more 'diverse' (i.e. discriminative) which seems to help the TL and MTL models learn the less frequent `PCEDT` relations.

For completeness, we investigate why the `PCEDT` functor `AIM` is never predicted even though it is more frequent than `TWHEN` (cf. Figure 1). We find that `AIM` is almost always misclassifed as `RSTR` by all the models. Furthermore, we discover that `AIM` and `RSTR` have the highest lexical overlap in the training set among all other pairs of relations in `PCEDT`: 78.35% of the left constituents and 73.26% of the right constituents of the compounds annotated as `AIM` occur in other

	`RSTR`	`PAT`	`REG`	`APP`	`ACT`	`AIM`	`TWHEN`
`A1`	0.70	0.90	0.78	0.62	0.47	0.65	0.10
`A2`	0.11	0.05	0.10	0.21	0.03	0.12	0.03
`A0`	0.06	0.01	0.04	0.13	0.47	0.07	-
`A3`	0.06	0.02	0.06	0.02	0.01	0.06	-
`LOC`	0.02	0.01	0.00	0.01	0.01	0.01	0.02
`TMP`	0.01	-	0.00	0.00	-	-	0.80
`MNR`	0.02	0.00	0.00	-	0.01	-	-
Count	3617	1312	777	499	273	116	59

Table 6: Correspondence matrix between `PCEDT` functors and `NomBank` arguments. Slots with '-' mean zero, 0.00 is a very small number but not zero.

	`A1`	`A2`	`A0`	`A3`	`LOC`	`TMP`	`MNR`
`RSTR`	0.51	0.54	0.47	0.63	0.66	0.36	0.78
`PAT`	0.24	0.09	0.03	0.08	0.07	-	0.05
`REG`	0.12	0.11	0.07	0.13	0.02	0.01	0.01
`APP`	0.06	0.14	0.13	0.03	0.05	0.01	-
`ACT`	0.03	0.01	0.26	0.01	0.03	-	0.03
`AIM`	0.02	0.02	0.02	0.02	0.01	-	-
`TWHEN`	0.00	0.00	-	-	0.01	0.46	-
Count	4932	715	495	358	119	103	79

Table 7: Correspondence matrix between `NomBank` arguments and `PCEDT` functors.

compounds annotated as `RSTR`. This explains why none of the models manage to learn the relation `AIM` but raises a question about the models' ability to learn relational representations; we pursue this question further in Section 7.3.

Finally, to clearly demonstrate the benefits of TL and MTL for `NomBank` and `PCEDT`, we report the F$_1$ macro-average scores in Table 8 (which is arguably the appropriate evaluation measure for imbalanced classification problems). Note that the relations that are not predicted by any of the models are not included in computing the macro-average. From Table 8 it becomes crystal clear that TL and MTL on the embedding layer yield remarkable improvements for `PCEDT` with about 7–

Model	NomBank	PCEDT
STL	52.66	40.15
TL_E	52.83	**48.34**
TL_H	52.98	46.52
TL_{EH}	**53.31**	47.12
MTL_E	53.21	47.23
MTL_F	42.07	40.73

Table 8: Macro-average F_1 score on the test split.

Model	NomBank			PCEDT		
	L	R	L&R	L	R	L&R
Count	351	286	72	351	286	72
STL	27.92	39.51	50.00	45.01	47.55	41.67
TL_E	25.93	36.71	48.61	**43.87**	47.55	41.67
TL_H	26.21	38.11	50.00	46.15	49.30	47.22
TL_{EH}	26.50	38.81	52.78	45.87	47.55	43.06
MTL_E	24.50	**33.22**	**38.89**	44.44	**47.20**	43.06
MTL_F	**22.79**	34.27	40.28	44.16	47.90	**38.89**

Table 9: Generalization error on the subset of unseen compounds in the test split. L: Left constituent. R: Right constituent. L&R: Completely unseen.

8 absolute points increase in macro-average F_1, in contrast to just 0.65 in the best case on NomBank.

7.3 Generalization on Unseen Compounds

Now we turn to analyze the models' ability to generalize over compounds unseen in the training split. Recent work by Dima (2016) and Vered and Waterson (2018) suggest that the gains achieved in noun–noun compound interpretation using word embeddings and somewhat similar neural classification models are in fact a by-product of *lexical memorization* (Levy et al., 2015); in other words, the classification models learn that a specific set of nouns is a strong indicator of a specific relation. Therefore, in order to gauge the role of lexical memorization in our models also, we quantify the number of unseen compounds that the STL, TL and MTL models predict correctly.

We distinguish between 'partly' and 'completely' unseen compounds. A compound is considered 'partly' unseen if one of its constituents (right or left) is not seen in the training data at all. A completely unseen compound is one whose left *and* right constituent are not seen in the training data (i.e. completely unseen compounds are the subset of compounds in the test split that have zero lexical overlap with the training split). Overall, almost 20% of the compounds in the test split have an unseen left constituent, about 16% of the compounds have unseen right constituent and 4% are completely unseen. In Table 9, we compare the performance of the different models on these three groups in terms of the proportion of compounds a model *misclassifies* in each group.

From Table 9 we see that TL and MTL reduce the NomBank generalization error in all cases, except TL_H and TL_{EH} on completely unseen compounds; the latter leads to higher generalization error. The MTL models lead to the biggest error reduction across the three types of unseen compounds; MTL_E leads to about six points error reduction on compounds with unseen right constituent and eleven points on completely unseen ones, and MTL_F reduces the error on unseen left constituent by five points. Note, however, that these results have to be read together with the *Count* row in Table 9 to get a complete picture. For instance, an eleven-point decrease in error on completely unseen compounds amounts to eight compounds. In PCEDT, the largest error reduction on unseen left constituents is 1.14 points which amounts to four compounds, 0.35 (just one compound) on unseen right constituents and 2.7 (or two compounds) on completely unseen compounds.

Since we see larger reductions in the generalization error in NomBank, we manually inspect the compounds that led to these reductions; i.e. we inspect the distribution of relations in the set of the correctly predicted unseen compounds. The MTL_E model reduces the generalization error on completely unseen compounds by a total of eight compounds compared to the STL model, but seven of these compounds are annotated with ARG1 which is the most frequent relation in NomBank. When it comes to the unseen right constituents, the 24 compounds MTL_E improves on consist of 18 ARG1 compounds, 5 ARG0 compounds and one ARG2 compound. We see a similar pattern upon inspecting the gains of the TL_E model; where most of the improvement arises from predicting more ARG1 and ARG0 correctly.

The majority of the partly or completely unseen compounds that were misclassified by all models are *not* of type ARG1 in NomBank or RSTR in PCEDT. This, together with the fact that the correctly predicted unseen compounds are annotated with the most frequent relations, indicate that the classification models rely on lexical memorization to learn the interpretation of compound relations.

Finally, to complement our understanding of the effect of lexical memorization, we plot the ratio of relation-specific constituents in NomBank and

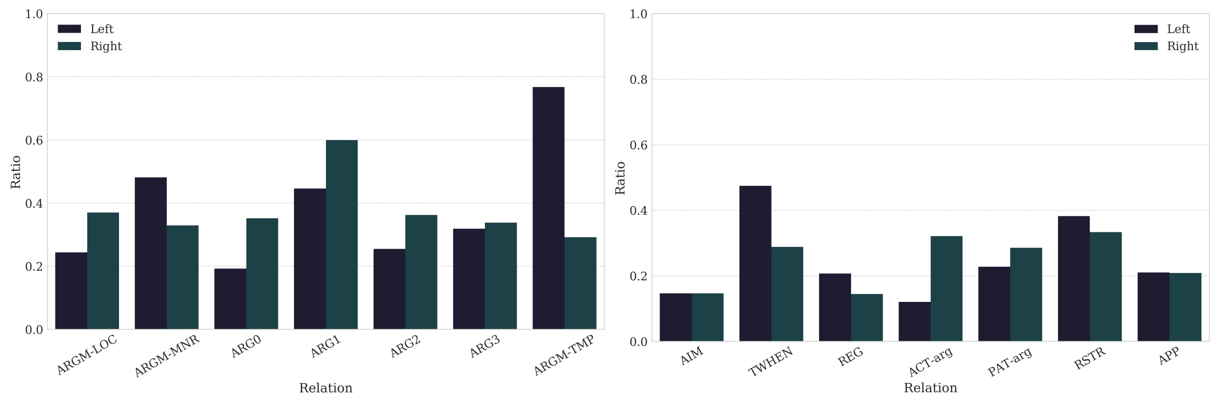

Figure 2: Ratio of relation-specific constituents in `NomBank` (left) and `PCEDT` (right).

`PCEDT` in Figure 2. We define relation-specific constituents as left or right constituents that only occur with one specific relation in the training split, and their ratio is simply their proportion in the overall set of left or right constituents per relation. Looking at Figure 2, we see that `NomBank` relations have higher ratios of relation-specific constituents in comparison to `PCEDT`, which arguably makes learning the former comparatively easier if the model is only to rely on lexical memorization. Furthermore, `ARGM-TMP` in `NomBank` and `TWHEN` in `PCEDT` stand out from other relations in Figure 2, which are also the two relations with the second highest F_1 score in their respective dataset—except in STL on `PCEDT` (cf. Tables 4 and 5). Lexical memorization is, therefore, the most likely explanation of such relatively high F_1 scores. We also observe some correlation between lower ratios of relation-specific constituents and relatively low F_1 scores, e.g. `APP` and `REG` in `PCEDT`. Based on these observations, we cannot rule out that our models exhibit some degree of lexical memorization effects, even though manual result analysis also reveals 'counter-examples' where the models generalize and make correct predictions where lexical memorization is impossible.

8 Conclusion

Transfer and multi-task learning for NLP currently receive a lot of attention, but for the time being there remains considerable uncertainty about which task properties and experimental settings actually are effective. In this work, we seek to shed light on the utility of TL and MTL perspectives on the semantic interpretation of noun–noun compounds. Through a comprehensive series of minimally contrasting experiments and in-depth analysis of results and prediction errors, we demonstrate the ability of both TL and MTL to mitigate the challenges of class imbalance and substantially improve prediction of low-frequency relations. In a nutshell, our TL and in particular MTL models make quantitatively and qualitatively better predictions, especially so on the 'hardest' inputs involving at least one constituent not seen in the training data—but clear indicators of remaining 'lexical memorization' effects arise from our error analysis of unseen compounds.

In general, transfer of representations or sharing across tasks is most effective at the embedding layers, i.e. the model-internal representation of the two compound constituents involved. In multi-task learning, full sharing of the model architecture across tasks worsens the model's ability to generalize on the less frequent relations.

We experience the dataset by Fares (2016) as an interesting opportunity for innovative neural approaches to compound interpretation, as it relates this sub-problem to broad-coverage semantic role labeling or semantic dependency parsing in `PCEDT` and `NomBank`. In future work, we plan to incorporate other NLP tasks defined over these frameworks to learn noun–noun compound interpretation using TL and MTL. Such tasks include semantic role labeling of nominal predicates in `NomBank` annotations as well as verbal predicates in `PropBank` (Kingsbury and Palmer, 2002).

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Associa-*

tion for Computational Linguistics: Volume 2, Short Papers, page 164 – 169, Valencia, Spain. Association for Computational Linguistics.

Lou Burnard. 2000. Reference guide for the British National Corpus version 1.0.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41 – 75.

Yu-An Chung, Hung-yi Lee, and James Glass. 2018. Supervised and Unsupervised Transfer Learning for Question Answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, page 1585 – 1594. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML*, page 160 – 167.

Corina Dima. 2016. On the Compositionality and Semantic Interpretation of English Noun Compounds. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, page 27 – 39. Association for Computational Linguistics.

Corina Dima and Erhard Hinrichs. 2015. Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings. In *Proceedings of the 11th International Conference on Computational Semantics*, page 173 – 183, London, UK. Association for Computational Linguistics.

Pamela Downing. 1977. On the Creation and Use of English Compound Nouns. *Language*, 53(4):810 – 842.

Murhaf Fares. 2016. A Dataset for Joint Noun-Noun Compound Bracketing and Interpretation. In *Proceedings of the ACL 2016 Student Research Workshop*, page 72 – 79. Association for Computational Linguistics.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, page 271 – 276, Gothenburg, Sweden. Association for Computational Linguistics.

Timothy Wilking Finin. 1980. The Semantic Interpretation of Nominal Compounds. In *Proceedings of the First Annual National Conf. on Artificial Intelligence*. AAAI Press.

Christina L. Gagné and Edward J. Shoben. 1997. Influence of Thematic Relations on the Comprehension of Modifier–Noun Combinations. *Journal of Experimental Psychology: Learning Memory and Cognition*, 23(1):71 – 87.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2009. Classification of semantic relations between nominals. *Language Resources and Evaluation*, 43(2):105 – 121.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, page 3153 – 3160, Istanbul, Turkey.

Su Nam Kim and Timothy Baldwin. 2005. Automatic Interpretation of Noun Compounds Using WordNet Similarity. In *Second International Joint Conference on Natural Language Processing*, page 945 – 956, Berlin, Heidelberg. Springer Berlin Heidelberg.

Su Nam Kim and Timothy Baldwin. 2013. A lexical semantic approach to interpreting and bracketing English noun compounds. *Natural Language Engineering*, 19(03):385 – 407.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, page 1989 – 1993, Las Palmas, Spain.

Mark Lauer. 1995. *Designing Statistical Language Learners. Experiments on Noun Compounds*. Doctoral dissertation, Macquarie University, Sydney, Australia.

Judith N Levi. 1978. *The syntax and semantics of complex nominals*. Academic Press.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 970 – 976, Denver, Colorado. Association for Computational Linguistics.

Charles Na Li. 1972. *Semantics and the Structure of Compounds in Chinese*. Ph.D. thesis, University of California, Berkeley.

Marco Marelli, Christina L. Gagné, and Thomas L. Spalding. 2017. Compounding as Abstract Operation in Semantic Space: Investigating relational effects through a large-scale, data-driven computational model. *Cognition*, 166:207 – 224.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, page 44 – 53, Valencia, Spain. Association for Computational Linguistics.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, page 803 – 806, Lisbon, Portugal.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, page 479 – 489, Austin, Texas. Association for Computational Linguistics.

Preslav Nakov. 2008. Noun Compound Interpretation Using Paraphrasing Verbs: Feasibility Study. In *Artificial Intelligence: Methodology, Systems, and Applications*, page 103 – 117, Berlin, Heidelberg. Springer Berlin Heidelberg.

Preslav Ivanov Nakov. 2007. *Using the Web as an Implicit Training Set: Application to Noun Compound Syntax and Semantics*. Doctoral dissertation, EECS Department, University of California, Berkeley.

Diarmuid Ó Séaghdha. 2007. Annotating and Learning Compound Noun Semantics. In *Proceedings of the ACL 2007 Student Research Workshop*, page 73 – 78, Prague, Czech Republic.

Diarmuid Ó Séaghdha. 2008. Learning Compound Noun Semantics. Technical Report UCAM-CL-TR-735, University of Cambridge, Computer Laboratory, Cambridge, UK.

Diarmuid Ó Séaghdha and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, page 621 – 629, Athens, Greece. Association for Computational Linguistics.

Diarmuid Ó Séaghdha and Ann Copestake. 2013. Interpreting compound nouns with kernel methods. *Journal of Natural Language Engineering*, 19(3):331 – 356.

Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345 – 1359.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07. Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Meeting of the Association for Computational Linguistics*, page 2037 – 2048, Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1532 – 1543, Doha, Qatar. Association for Computational Linguistics.

Lutz Prechelt. 2012. Early stopping — but when? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mary Ellen Ryder. 1994. *Ordered chaos: The interpretation of English noun-noun compounds*, volume 123. University of California Press.

Vered Shwartz and Ido Dagan. 2018. Paraphrase to Explicate: Revealing Implicit Noun-Compound Relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 1200 – 1211. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 231 – 235. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, page 678 – 687, Uppsala, Sweden. Association for Computational Linguistics.

Shwartz Vered and Chris Waterson. 2018. Olive Oil is Made *of* Olives, Baby Oil is Made *for* Babies: Interpreting Noun Compounds using Paraphrases in a Neural Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, page 218 – 224, New Orleans, Louisiana. Association for Computational Linguistics.