When is multitask learning effective? Semantic sequence prediction under varying data conditions

Héctor Martínez Alonso[♠] **Barbara Plank**[♡]
[♡] Center for Language and Cognition, University of Groningen, The Netherlands
[♠] Univ. Paris Diderot, Sorbonne Paris Cité – Alpage, INRIA, France
hector.martinez-alonso@inria.fr,b.plank@rug.nl

Abstract

Multitask learning has been applied successfully to a range of tasks, mostly morphosyntactic. However, little is known on *when* MTL works and whether there are data characteristics that help to determine its success. In this paper we evaluate a range of semantic sequence labeling tasks in a MTL setup. We examine different auxiliary tasks, amongst which a novel setup, and correlate their impact to data-dependent conditions. Our results show that MTL is not always effective, significant improvements are obtained only for 1 out of 5 tasks. When successful, auxiliary tasks with compact and more uniform label distributions are preferable.

1 Introduction

The recent success of recurrent neural networks (RNNs) for sequence prediction has raised a great deal of interest, which has lead researchers to propose competing architectures for several language-processing tasks. These architectures often rely on multitask learning (Caruana, 1997).

Multitask learning (MTL) has been applied with success to a variety of sequence-prediction tasks including chunking and tagging (Collobert et al., 2011; Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank, 2016), name error detection (Cheng et al., 2015) and machine translation (Luong et al., 2016). However, little is known about MTL for tasks which are more *semantic* in nature, i.e., tasks that aim at labeling some aspect of the meaning of words (Cruse, 1986), instead their morphosyntactic behavior. In fact, results on semantic tasks are either mixed (Collobert et al., 2011) or, due to the file drawer bias (Rosenthal, 1979), simply not reported. There is no prior study—to the best of our knowledge—that compares data-dependent conditions with performance measures to shed some light on when MTL works for semantic sequence prediction. Besides any variation in annotation and conceptualization, the label distributions of such semantic tasks tends to be very different to the characteristic distributions expected in more frequently studied morphosyntactic tasks such as POS-tagging.

The main contribution of this work is an evaluation of MTL on semantic sequence prediction on data-dependent conditions. We derive characteristics of datasets that make them favorable for MTL, by comparing performance with information-theoretical metrics of the label frequency distribution.

We use an off-the-shelf state-of-the-art architecture based on bidirectional Long-Short Term Memory (LSTM) models (Section 3) and evaluate its behavior on a motivated set of main and auxiliary tasks. We gauge the performance of the MTL setup (Section 4) in the following ways: i) we experiment with different combinations of main and auxiliary tasks, using semantic tasks as main task and morphosyntactic tasks as auxiliary tasks; ii) we apply FREQBIN, a frequency-based auxiliary task (see Section 2.5) to a series of language-processing tasks and evaluate its contribution, and iii) for POS we experiment with different data sources to control for label inventory size and corpus source for the auxiliary task.

From our empirical study we observe the MTL architecture's sensitivity to label distribution properties, and its preference for compact, mid-entropy distributions. Additionally, we provide a novel parametric refinement of the FREQBIN auxiliary task that is more robust. In broader terms, we expect to motivate more thorough analysis of the performance of neural networks in MTL setups.

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 44–53,
Valencia, Spain, April 3-7, 2017.

2 Analyzing multi-task learning

Multitask learning systems are often designed with the intention of improving a *main* task by incorporating joint learning of one or more related *auxiliary* tasks. For example, training a MTL model for the main task of chunking and treating part-of-speech tagging (POS) as auxiliary task.

The working principle of multitask learning is to improve generalization performance by leveraging training signal contained in related tasks (Caruana, 1997). This is typically done by training a single neural network for multiple tasks jointly, using a representation that is shared across tasks. The most common form of MTL is the inclusion of one output layer per additional task, keeping all hidden layers common to all tasks. Task-specific output layers are customarily placed at the outermost layer level of the network.

In the next section, we depict all main and auxiliary tasks considered in this paper.

2.1 Main tasks

We use the following main tasks, aimed to represent a variety of semantic sequence labeling tasks.
FRAMES: We use the FrameNet 1.5 (Baker et al., 1998) annotated corpus for a joint frame detection and frame identification tasks where a word can receive a predicate label like *Arson* or *Personal success*. We use the data splits from (Das et al., 2014; Hermann et al., 2014). While frame identification is normally treated as single classification, we keep the sequence-prediction paradigm so all main tasks rely on the same architecture.
SUPERSENSES: We use the supersense version of SemCor (Miller et al., 1993) from (Ciaramita and Altun, 2006), with coarse-grained semantic labels like *noun.person* or *verb.change*.
NER: The CONLL2003 shared-task data for named entity recognition for labels *Person*, *Loc*, etc. (Tjong Kim Sang and De Meulder, 2003).
SEMTRAITS: We have used the EurWordNet list of ontological types for senses (Vossen et al., 1998) to convert the SUPERSENSES into coarser semantic traits like *Animate* or *UnboundedEvent*.[1]
MPQA: The Multi-Perspective Question Answering (MPQA) corpus (Deng and Wiebe, 2015), which contains sentiment information among others. We use the annotation corresponding to the *coarse* level of annotation, with labels like *attitude* and *direct-speech-event*.

[1]Available at: https://github.com/bplank/multitasksemantics

2.2 Auxiliary tasks

We have chosen auxiliary tasks that represent the usual features based on frequency and morphosyntax used for prediction of semantic labels. We collectively refer to them as *lower-level tasks*.
CHUNK: The CONLL2003 shared-task data for noun- and verb-phrase chunking (Tjong Kim Sang and De Meulder, 2003).
DEPREL: The dependency labels for the English Universal Dependencies v1.3 (Nivre et al., 2016).
FREQBIN: The log frequency of each word, treated as a discrete label, cf. Section 2.5.
POS: The part-of-speech tags for the Universal Dependencies v1.3 English treebank.

2.3 Data properties

Table 1 lists the datasets used in this paper, both to train main tasks and auxiliary tasks. For each dataset we list the following metrics: number of sentences, number of tokens, token-type ratio (TTR), the size of the label inventory counting B-labels and I-labels as different ($|Y|$), and the proportion of out-of-span labels, which we refer to as O labels.

The table also provides some of the information-theoretical measures we describe in Section 2.4. Note that DEPRELS and POS are the only datasets without any O labels, while FRAMES and SEMTRAITS are the two tasks with O labels but no B/I-span notation, as tokens are annotated individually.

2.4 Information-theoretic measures

In order to quantify the properties of the different label distributions, we calculate three information-theoretical quantities based on two metrics, kurtosis and entropy.

Entropy is the best-known information-theoretical metric. It indicates the amount of uncertainty in a distribution. We calculate two variants of entropy, one taking all labels in consideration $\mathrm{H}(\mathrm{Y}_{full})$, and another one $\mathrm{H}(\mathrm{Y}_{-O})$ where we discard the O label and only measure the entropy for the named labels, such as frame names in FRAMES. The entropy of the label distribution $\mathrm{H}(\mathrm{Y}_{full})$ is always lower than the entropy for the distribution disregarding the O label $\mathrm{H}(\mathrm{Y}_{-O})$. This difference is a consequence

	sentences	tokens	TTR	$\|Y\|$	prop of O	k(Y)	H(Y_{full})	H(Y_{-O})
FRAMES	5.9k	119k	.12	707	.80	701.41	1.60	5.51
MPQA	1.7k	44k	.15	9	.65	2.79	1.12	1.33
NER	22.1k	303k	.10	9	.83	4.10	0.77	1.93
SEMTRAITS	20k	435k	.07	11	.66	5.68	1.29	1.89
SUPERSENSES	20k	435k	.07	83	.66	76.73	1.84	3.53
CHUNK	22.1k	303k	.10	22	.14	3.68	1.73	1.54
DEPRELS	16.6k	255k	.09	47	-	1.80	3.11	3.11
FREQBIN	*Same as respective main task*			4–7	-	*Depends on variant*		
POS	16.6k	255k	.09	17	-	-0.20	2.49	2.49

Table 1: Datasets for main tasks (above) and auxiliary tasks (below) with their number of sentences, tokens, type-token ratio, size of label inventory, proportion of O labels, kurtosis of the label distribution, entropy of the label distribution, and entropy of the label distribution without the O label.

of the O-label being often the majority class in span-annotated datasets. The only exception is CHUNK, where O-tokens make up 14% of the total, and the full-distribution entropy is higher.

Kurtosis indicates the skewness of a distribution and provides a complementary perspective to the one given by entropy. The kurtosis of the label distribution describes its tailedness, or lack thereof. The kurtosis for a normal distribution is 3, and higher kurtosis values indicate very tailed distributions, while lower kurtosis values indicate distributions with fewer outliers.

For instance, we can see that larger inventory sizes yield more heavy-tailed distributions, e.g. FRAMES presents a lot of outliers and has the highest kurtosis. The very low value for POS indicates a distribution that, although Zipfian, has very few outliers as a result of the small label set. In contrast, DEPRELS, coming from the same corpus, has about three times as many labels, yielding a distribution that has fewer mid-values while still being less than 3. Nevertheless, the entropy values of POS and DEPRELS are similar, so kurtosis provides a complementary perspective on the data.

2.5 FREQBIN variants

Recently, a simple auxiliary task has been proposed with success for POS tagging: predicting the log frequency of a token (Plank et al., 2016). The intuition behind this model is that the auxiliary loss, predicting word frequency, helps differentiate rare and common words, thus providing better predictions for frequency-sensitive labels. They refer to this auxiliary task as FREQBIN, however, focus on POS only. Plank et al. (2016) used the discretized log frequency of the current word to build the FREQBIN auxiliary task to aid POS tagging, with good results. This auxiliary task aids the prediction of the main task (POS) in about half the languages, and improves the prediction of out of vocabulary words. Therefore, it is compelling to assess the possible contribution of FREQBIN for other tasks, as it can be easily calculated from the same training data as the main task, and requires no external resources or annotation.

We experiment with three different variants of FREQBIN, namely:

1. SKEWED$_{10}$: The original formulation of $a = int(log_{10}(freqtrain(w))$, where a is the frequency label of the word w. Words not in the training data are treated as hapaxes.
2. SKEWED$_5$: A variant using 5 as logarithm base, namely $a = int(log_5(freqtrain(w))$, aimed at providing more label resolution, e.g. for the NER data, SKEWED$_{10}$ yields 4 different labels, and SKEWED$_5$ yields 6.
3. UNIFORM: Instead of binning log frequencies, we take the index of the k-quantilized cumulative frequency for a word w. We use this parametric version of FREQBIN with the median number of labels produced by the previous variants to examine the importance of the label distribution being skewed. For k=5, this variant maximizes the entropy of a FREQBIN five-label distribution. Note that this method still places all hapaxes and out-of-vocabulary words of the test data in the same frequency bin.

Even though we could have used a reference corpus to have the same FREQBIN for all the data, we prefer to use the main-task corpus for FREQBIN. Using an external corpus would otherwise lead to a semisupervised learning scenario which is out of the scope of our work. Moreover, in us-

ing only the input corpus to calculate frequency we replicate the setup of Plank et al. (2016) more closely.

3 Model

Recurrent neural networks (RNNs) (Elman, 1990; Graves and Schmidhuber, 2005) allow the computation of fixed-size vector representations for word sequences of arbitrary length. An RNN is a function that reads in n vectors $x_1, ..., x_n$ and produces a vector h_n, that depends on the entire sequence $x_1, ..., x_n$. The vector h_n is then fed as an input to some classifier, or higher-level RNNs in stacked/hierarchical models. The entire network is trained jointly such that the hidden representation captures the important information from the sequence for the prediction task.

A bi-directional recurrent neural network (Graves and Schmidhuber, 2005) is an extension of an RNN that reads the input sequence twice, from left to right and right to left, and the encodings are concatenated. An LSTM (Long Short-Term Memory) is an extension of an RNN with more stable gradients (Hochreiter and Schmidhuber, 1997). Bi-LSTM have recently successfully been used for a variety of tasks (Collobert et al., 2011; Huang et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015; Kiperwasser and Goldberg, 2016; Liu et al., 2015; Plank et al., 2016). For further details, cf. Goldberg (2015) and Cho (2015).

We use an off-the-shelf bidirectional LSTM model (Plank et al., 2016).[2] The model is illustrated in Figure 1. It is a context bi-LSTM taking as input word embeddings $\vec{w}$. Character embeddings $\vec{c}$ are incorporated via a hierarchical bi-LSTM using a sequence bi-LSTM at the lower level (Ballesteros et al., 2015; Plank et al., 2016). The character representation is concatenated with the (learned) word embeddings $\vec{w}$ to form the input to the context bi-LSTM at the upper layers. For hyperparameter settings, see Section 3.1.

The stacked bi-LSTMs represent the shared layers between tasks. We here use three stacked (h=3) bi-LSTMs for the upper layer, and a single layer bi-LSTM at the lower level for the character representations. Following Collobert et al. (2011), at the outermost ($h = 3$) layer separate output layers for the single tasks are added using a softmax. We additionally experiment with predicting lower-level tasks at inner layers, i.e., predicting POS at $h = 1$, while the main task at $h = 3$, the outermost layer, following Søgaard and Goldberg (2016). During training, we randomly sample a task and instance, and backpropagate the loss of the current instance through the shared deep network. In this way, we learn a joint model for main and auxiliary task(s).

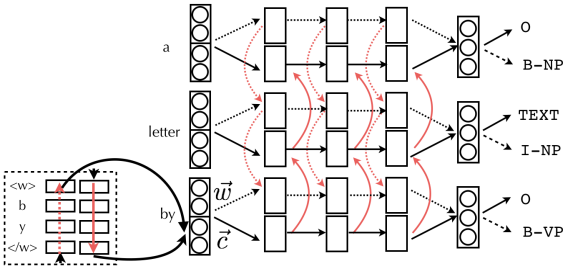

Figure 1: Multi-task bi-LSTM. The input to the model are word $\vec{w}$ and character embeddings $\vec{c}$ (from the lower bi-LSTM). The model is a stacked 3-layer bi-LSTM with separate output layers for the main task (solid line) and auxiliary tasks (dashed line; only one auxiliary task shown in the illustration).

3.1 Hyperparameters

All the experiments in this article use the same bi-LSTM architecture described in Section 3. We train the bi-LSTM model with default parameters, i.e., SGD with cross-entropy loss, no mini-batches, 30 epochs, default learning rate (0.1), 64 dimensions for word embeddings, 100 for character embeddings, 100 hidden states, random initialization for the embeddings, Gaussian noise with σ=0.2. We use a fixed random seed set upfront to facilitate replicability. The only hyperparameter we further examine is the number of epochs, which is set to 30 unless otherwise specified.

We follow the approach of Collobert et al. (2011) in that we do *not* use any task-specific features beyond word and character information, nor do we use pre-trained word embeddings for initialisation or more advanced optimization techniques.[3] While any of these changes would likely improve the performance of the systems, the goal of our experiments is to delimit the behavior of the bi-LSTM architecture and the interaction between main and auxiliary task(s).

[2] Available at: `https://github.com/bplank/bilstm-aux`

[3] For example, `AdamTrainer` or `MomentumSGDTrainer` in `pycnn`.

3.2 Experimental Overview

A system in our experiments is defined by a main task and up to two auxiliary tasks, plus a choice of output layers (at which layer to predict the auxiliary task, i.e., $h \in \{1,2,3\}$). For each main task, we ran the following systems:

1. Baseline, without any auxiliary task.
2. One additional system for each auxiliary task, say DEPREL.
3. A combination of each of the three versions of FREQBIN, namely SKEWED_5,SKEWED_{10} and UNIFORM, and each of the other auxiliary tasks, such as DEPREL+UNIFORM.

The total combination of systems for all five main tasks is 1440.

4 Results

This section describes the results of both experimental scenarios, namely the benchmarking of FREQBIN as an auxiliary task, and the combinations of semantic main task with low-level auxiliary tasks, including an analysis of the data properties. The different tasks in our experiments typically use different evaluation metrics, however we evaluate all tasks on micro-averaged F1 without the O class, which we consider the most informative overall. We do not use the O-label's F1 score because it takes recall into consideration, and it is deceptively high for the majority class. We test for significance with a 10K-iteration bootstrap sample test, and $p < .05$.

4.1 Main semantic tasks

This section presents the results for the prediction of the main semantic tasks described in Section 2. Given the size of the space of possible task combinations for MTL, we only report the baseline and the results of the best system. Table 2 presents the results for all main semantic tasks, comparing the results of the best system with the baseline. The last column indicates the amount of systems that beat the baseline for a given certain main task. Having fixed the variant of FREQBIN to UNIFORM (see Section 4.2), and the number of epochs to 30 (see below) on development data, the total amount of systems for any main task is 22.

Out of the two main tasks over the baseline only SEMTRAITS is significantly better over BL. SEMTRAITS has a small label set, so the system is able to learn shared parameters for the label combinations of main and aux without suffering from too much sparsity. Compare with the dramatic loss of the already low-performing FRAMES, which has the highest kurtosis caused by the very long tail of low-frequency labels.

	BL	ΔBest	Description	aux layer	# over
FRAMES	38.93	-8.13	+FREQBIN	outer	0
MPQA	28.26	**0.96**	+POS+FREQBIN	inner	2
NER	90.60	-0.58	+FREQBIN	inner	0
SEMTRAITS	70.42	**1.24**	+FREQBIN	outer	13
SUPERSENSES	62.36	-0.13	+POS+FREQBIN	inner	0

Table 2: Baseline (BL) and best system performance difference (Δ) for all main tasks—improvements in bold, significant improvements underlined—plus number of systems over baseline for each main task.

We have expected CHUNK to aid SUPERSENSES, but in spite of our expectations, other low-level tasks do not aid in general the prediction of high-level task. What is otherwise an informative feature for a semantic task in single-task learning does not necessarily lend itself as an equally useful auxiliary task for MTL.

For a complementary evaluation, we have also measured the precision of the O label. However, precision score is also high, above 90, for all tasks except the apparently very difficult MPQA (70.41 for the baseline). All reported systems degrade around 0.50 points with regards to the baseline, except SUPERSENSES which improves slightly form 96.27 to 96.44. The high precision obtained for the also very difficult FRAMES tasks suggests that this architecture, while not suitable for frame disambiguation, can be used for frame-target identification. Disregarding FREQBIN, the only low-level tasks that seems to aid prediction is POS.

An interesting observation from the BIO task analysis is that while the standard bi-LSTM model used here does not have a Viterbi-style decoding like more complex systems (Ma and Hovy, 2016; Lample et al., 2016), we have found very few invalid BIO sequences. For NER, there are only ten I-labels after an O-label, out of the 27K predicted by the bi-LSTM. For SUPERSENSES there are 59, out of 1,5K predicted I-labels.

The amount of invalid predicted sequences is lower than expected, indicating that an additional decoding layer plays a smaller role in prediction quality than label distribution and corpus size, e.g. NER is a large dataset with few labels, and the system has little difficulty in learning label precedences. For larger label sets or smaller data sizes,

invalid sequence errors are bound to appear because of sparseness.

Effect of output layer choice We observe no systematic tendency for an output layer to be a better choice, and the results of choosing the inner- or outer-layer (h=1 vs h=3) input differ only minimally. However, both systems that include POS have a preference for the inner layer having higher performance, which is consistent with the results for POS in (Søgaard and Goldberg, 2016).

Effect of the number of training epochs Besides all the data properties, the only hyperparameter that we examine further is the number of network training epochs.[4] All the results reported in this article have been obtained in a 30-epoch regime. However, we have also compared system performance with different numbers of epochs. Out of the values we have experimented (5,15,30,50) with, we recommend 30 iterations for this architecture. At 5 and 15 epochs, the performance does not reach the levels for 30 and is consistently worse for baselines and auxiliary-task systems. Moreover, the performance for 50 is systematically worse than for 30, which indicates overfitting at this point.

Effect of training data size We have run all systems increasing the size of the main task training data in blocks of 25%, keeping the size of the auxiliary task constant. We do not observe improvements over baseline along the learning curve for any of the main tasks except MPQA and SEMTRAITS. At smaller main task data sizes, the auxiliary task learning swamps the training of the main task. This results is consistent with the findings by Luong et al. (2016). We leave the research on the effects auxiliary data size—and its size ratio with regards to the main task—for further work.

4.2 Auxiliary task contribution

As follows from the results so far, the bi-LSTM will not benefit from auxiliary loss if there are many labels and entropy is too high. Auxiliary task level distribution also plays a role, as we will discuss in Section 4.3, FREQBIN-UNIFORM consistently outperforms the skewed measure with base 5 and 10.

[4]Number of epochs is among the most influential parameters of the system. Adding more layers did not further improve results.

	BL	ΔUD/UPOS	ΔUD/PTB	ΔWSJ/PTB
FRAMES	38.93	-14.64	-16.02	-28.18
NER	90.60	-1.36	-2.05	-2.56
MPQA	28.26	-5.62	-13.53	-14.81
SEMTRAITS	70.42	0.67	-0.3	-0.14
SUPERSENSES	62.36	-2.86	-2.83	-6.32
CHUNK	94.76	0.2	0.18	0.18
DEPRELS	88.70	-0.19	-0.18	-1.06
POS	94.36	–	0.18	-0.53

Table 3: Comparison different POS variants (data source/tag granularity): Baseline (BL) and the difference in performance on the +POS system when using the UD Corpus with UPOS (UD/UPOS) or with PTB tabs (UD/PTB), as well as the Wall Street Journal with PTB tags (WSJ/PTB).

Therefore we have also measured the effect of using different sources of POS auxiliary data to give account for the possible differences in label inventory and corpus for all tasks, high and low-level, cf. Table 3. The English UD treebank is distributed with Universal POS (UPOS), which we use throughout this article, and also with Penn Treebank (PTB) tags (Marcus et al., 1993). We have used the PTB version of the English UD corpus (UD/PTB) as well as the training section of the Wall Street Journal (WSJ) treebank as of POS (WSJ/PTB) auxiliary task. The former offers the opportunity to change the POS inventory to the three times larger PTB inventory while using the same corpus.

However, the characteristics of the UD/UPOS we have used as POS throughout the article makes it a more suitable auxiliary source, in fact it systematically outperforms the other two. We argue that UD/UPOS has enough linguistic signal to be a useful auxiliary task, while still depending on a smaller label inventory. Interestingly, if we use POS for CHUNK (cf. Table 3), note that even though the language in WSJ is closer to the language in the training corpora for CHUNK and NER, it is not the best auxiliary POS source for either task.

We observe an improvement when using UD/PTB for POS, while using WSJ/PTB worsens the results for this task. We argue that this architecture benefits from the scenario where the same corpus is used to train with two different label sets for POS, whereas using a larger label set and a different corpus does not aid prediction.

4.3 Analyzing FREQBIN

In this section we evaluate the interaction between all tasks and the FREQBIN auxiliary task. For this purpose, we treat all tasks (high- or low-level) as main task, and compare the performance of a single-task baseline run, with a task +FREQBIN setup. We have compared the three versions of FREQBIN (Section 2.5) but we only report UNIFORM, which consistently outperforms the other two variants, according to our expectations.

Table 4 lists all datasets with the size of their label inventory for reference ($|Y|$), as well as the absolute difference in performance between the FREQBIN-UNIFORM system and the baseline (Δ). Systems that beat the baseline are marked in bold.

Following Plank et al. (2016), the FREQBIN system beats the baseline for the POS task. Moreover, it also aids the prediction for SEMTRAITS and MPQA. The better performance of these two systems indicates that this architecture is not necessarily only advisable for lower-level tasks, as long as the datasets have the right data properties.

	$\|Y\|$	BL	ΔU	R^2
FRAMES	707	38.93	-8.13	.00
MPQA	9	28.26	**0.44**	.09
NER	9	90.60	-1.31	.26
SEMTRAITS	11	70.42	**1.12**	.44
SUPERSENSES	83	62.36	-0.69	.47
CHUNK	22	94.76	-0.14	.49
POS	17	94.35	**0.21**	.68
DEPRELS	47	88.70	-0.16	.64

Table 4: Label inventory size ($|Y|$), FREQBIN-baseline absolute difference in performance (Δ)– improvements are in bold, significant improvements are underlined—and coefficient of determination for label-to-frequency regression (R^2).

The improvement of low-level classes is clear in the case of POS. We observe an improvement from 75 to 80 for the X label, mostly made up of low-frequency items. The similarly scattered label INTJ goes from 84 to 87. While no POS label drops in performance on +FREQBIN with regards to the baseline, all the other improvements are of 1 point of less.

4.4 Label–frequency co-informativeness

To supplement the benchmarking of FREQBIN, we estimate how much frequency information is contained in all the linguistic sequence annotations used in this article. We do so by evaluating the coefficient of determination (R^2) of a linear regression model to predict the log frequency of a word given its surrounding label trigram, which we use as a proxy for sequence prediction. For instance, for 'the *happy* child', it would attempt to predict the log-frequency of *happy* given the 'DET ADJ NOUN' POS trigram. Note that this model is delexicalized, and only uses task labels because its goal is to determine how much word-frequency information is contained in e.g. the POS sequence. A high R^2 indicates there is a high proportion of the variance of log frequency explained by the label trigram. We use linear regression implemented in `sklearn` with L2 regularization and report the average R^2 of 10-fold cross-validation.

POS is the label set with the highest explanatory power over frequency, which is expectable: determiners, punctuations and prepositions are high-frequency word types, whereas hapaxes are more often closed-class words. DEPRELS sequences contain also plenty of frequency information. Three sequence tasks have similar scores under .50, namely CHUNK, SUPERSENSE and SEMTRAITS. They all have in common that their O class is highly indicative of function words, an argument supported by their similar values of full-distribution entropy. The one with the lowest score out of these three, namely SEMTRAITS is the one with the least grammatical information, as it does not contain part of speech-related labels. The ($R2$) is very low for the remaining tasks, and indeed, for FRAMENET it is a very small negative number which rounds up to zero.

While the co-informativeness of FREQBIN with regards to its main task is a tempting explanation, it does not fully explain when it works as an auxiliary task. Indeed, the FREQBIN contribution at handling out-of-vocabulary words seems to only affect POS and SEMTRAITS, while it does not improve DEPRELS, which normally depends on syntactic trees for accurate prediction.

5 Net capacity and contribution of character representation

In this section we alter the network to study the effect of network width and character representations. Multitask learning allows easy sharing of parameters for different tasks. Part of the explanation for the success of multitask learning are related to *net capacity* (Caruana, 1997). Enlarg-

ing a network's hidden layers reduces generalization performance, as the network potentially learns dedicated parts of the hidden layer for different tasks. This means that the desirable trait of parameter sharing of MTL is lost. To test this property, we train a MTL network for all setups where we increase the size of the hidden layer by a factor k, where k is the number of auxiliary tasks.

Our results confirm that increasing the size of the hidden layers reduces generalization performance. This is the case for all setups. None of the results is better than the best systems in Table 2, and the effective number of systems that outperform the baseline are fewer (FRAMES 0, MPQA: 2, NER: 0, SEMTRAITS: 9, SUPERSENSES: 0).

Throughout the article we used the default network structure which includes a lower-level bi-LSTM at the character level. However, we hypothesize that the character features are not equally important for all tasks. In fact, if we disable the character features, making the system only depend on word information (cf. Table 5), we observe that two of the tasks (albeit the ones with the overall lowest performance) increase their performance in about 2.5 points, namely MPQA and FRAMES. For the other two tasks we observe drops up to a maximum of 8-points for NER. Character embeddings are informative for NER, because they approximate the well-known capitalization features in traditional models. Character features are not informative for tasks that are more dependent on word identity (like FRAMES), but are indeed useful for tasks where parts of the word can be informative, such as POS or NER.

	BL ($w + c$)	Δonly w
FRAMES	38.93	+2.39
NER	90.60	-8.05
MPQA	28.26	+2.91
SEMTRAITS	70.42	-3.62
SUPERSENSES	62.36	-4.44
CHUNK	94.76	-0.96
DEPRELS	88.70	-1.87
POS	94.36	-3.18

Table 5: Comparison default hierarchical systems using a lower-level bi-LSTM for characters (BL $w + c$) versus system using only words (w).

6 Related Work

Multitask learning has been recently explored by a number of studies, including name error recognition (Cheng et al., 2015), tagging and chunking (Collobert et al., 2011; Plank et al., 2016), entity and relation extraction (Gupta et al., 2016), machine translation (Luong et al., 2016) and machine translation quality estimation including modeling annotator bias (Cohn and Specia, 2013; Shah and Specia, 2016). Most earlier work had in common that it assumed jointly labeled data (same corpus annotated with multiple labels). In contrast, in this paper we evaluate multitask training from distinct sources to address data paucity, like done recently (Kshirsagar et al., 2015; Braud et al., 2016; Plank, 2016).

Sutton et al. (2007) demonstrate improvements for POS tagging by training a joint CRF model for both POS tagging and noun-phrase chunking. However, it is not clear under what conditions multi-task learning works. In fact, Collobert et al. (2011) train a joint feedforward neural network for POS, chunks and NER, and observe only improvements in chunking (similar to our findings, cf. Section 4.2), however, did not investigate data properties of these tasks.

To the best of our knowledge, this is the first extensive evaluation of the effect of data properties and main-auxiliary task interplay in MTL for semantic sequence tasks. The most related work is Luong et al. (2016), who focus on the effect of auxiliary data size (constituency parsing) on the main task (machine translation), finding that large amounts of auxiliary data swamp the learning of the main task. Earlier work related to MTL is the study by Ando and Zhang (2005) who learn many auxiliary task from unlabeled data to aid morphosyntactic tasks.

7 Conclusions and Future Work

We have examined the data-conditioned behavior of our MTL setup from three perspectives. First, we have tested three variants of FREQBIN showing that our novel parametric UNIFORM variant outperforms the previously used SKEWED$_{10}$, which has a number of labels determined by the corpus size. Second, we examined main-auxiliary task combinations for five semantic tasks and up to two lower-level tasks. We observe that the best auxiliary task is either FREQBIN or FREQBIN+POS, *which have low kurtosis and fairly high entropy*.

We also explored three sources of POS data as auxiliary task, differing in corpus composition or

label inventory. We observe that the UPOS variant is the most effective auxiliary task for the evaluated architecture. Indeed, UPOS has fewer labels, and also a more compact distribution with lower kurtosis than its PTB counterpart.

While we propose a better variant of FREQBIN (UNIFORM) we conclude that it is not a useful auxiliary task in the general case. Rather, it helps predict low-frequency labels in scenarios where the main task is already very co-informative of word frequency. While log frequency lends itself naturally to a continuous representation so that we could use regression to predict it instead of classification, doing so would require a change of the architecture and, most importantly, the joint loss. Moreover, discretized frequency distributions allow us to interpret them in terms of entropy. Thus, we leave it to future work.

When comparing system performance to data properties, we determine the architecture's preference for compact, mid-entropy distributions what are not very skewed, i.e., have low kurtosis. This preference explains why the system fares consistently well for a lot of POS experiments but falls short when used for task with many labels or with a very large O majority class. Regarding output layer choice, we have not found a systematic preference for inner or outer-layer predictions for an auxiliary task, as the results are often very close.

We argue strongly that the difficulty of semantic sequence predictions can be addressed as a matter of data properties and not as the antagonic truism that morphosyntax is easy and semantics is hard. The underlying problems of semantic task prediction have often to do with the skewedness of the data, associated often to the preponderance of the O-class, and a possible detachment from mainly lexical prediction, such as the spans of MPQA.

This paper is only one step towards better understanding of MTL. It is necessarily incomplete, we hope to span more work in this direction. For instance, the system evaluated in this study has no Viterbi-style decoding for sequences. We hypothesize that such extension of the model would improve prediction of labels with strong interdependency, such as BIO-span labels, in particular for small datasets or large label inventories, albeit we found the current system predicting fewer invalid sequences than expected. In future, we would like to extend this work in several directions: comparing different MTL architectures, additional tasks, loss weighting, and comparing the change of performance between a label set used as an auxiliary task or as a—predicted—feature.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback. Barbara Plank thanks the Center for Information Technology of the University of Groningen for the HPC cluster and Nvidia corporation for supporting her research. Héctor Martínez Alonso is funded by the French DGA project VerDi.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *ACL*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *EMNLP*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *COLING*.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of rst discourse parsers. In *COLING*.

Rich Caruana. 1997. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-Domain Name Error Detection using a Multi-Task RNN. In *EMNLP*.

Kyunghyun Cho. 2015. Natural Language Understanding with Distributed Representation. *ArXiv*, abs/1511.07916.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *ACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

D. Alan Cruse. 1986. *Lexical semantics*. Cambridge University Press.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *NAACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Yoav Goldberg. 2015. A Primer on Neural Network Models for Natural Language Processing. *ArXiv*, abs/1510.00726.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction. In *COLING*.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *TACL*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *ACL-IJCNLP*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *ACL*.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *COLING*.

Robert Rosenthal. 1979. The file drawer problem and tolerance for null results. *Psychological bulletin*, 86(3):638.

Kashif Shah and Lucia Specia. 2016. Large-scale multitask learning for machine translation quality estimation. In *NAACL*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*, pages 142–147. Association for Computational Linguistics.

Piek Vossen, Laura Bloksma, Horacio Rodriguez, Salvador Climent, Nicoletta Calzolari, Adriana Roventini, Francesca Bertagna, Antonietta Alonge, and Wim Peters. 1998. The eurowordnet base concepts and top ontology. *Deliverable D017 D*, 34:D036.