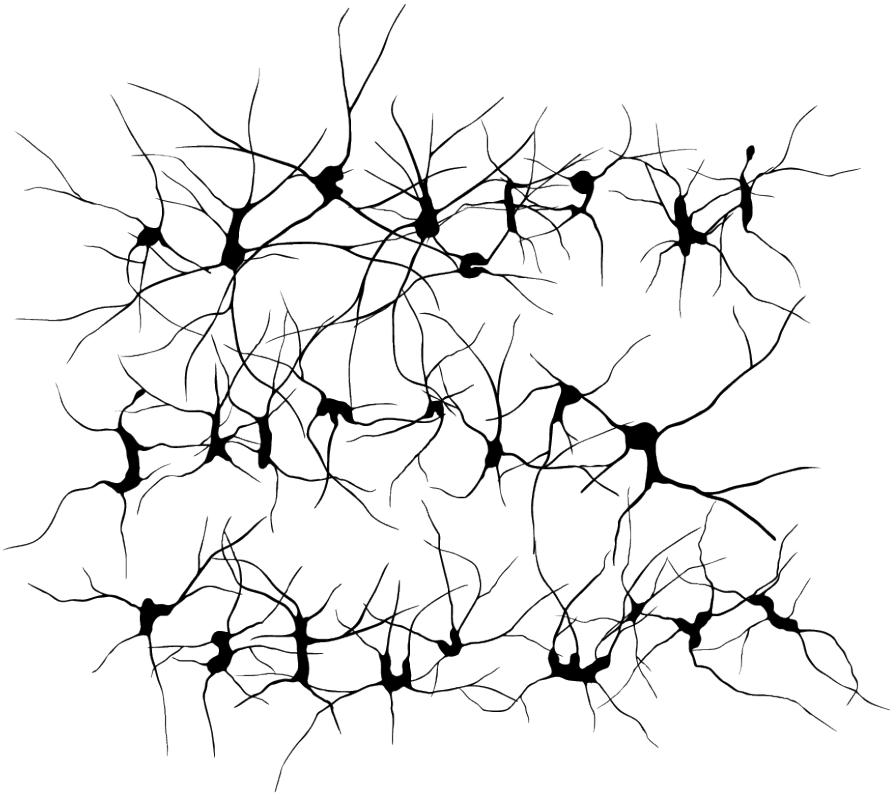
One Model to Rule them All

Bjerva, Johannes

PART II

Multitask Learning



CHAPTER 4

*Multitask Semantic Tagging with Residual Networks

Abstract | In this chapter, a semantic tag set is presented, which is tailored for multilingual semantic parsing. As a first step towards exploring multitask learning, we will look at the effects of jointly learning this task, together with POS tagging, compared to learning the two tasks separately. Furthermore, we will see a deep neural network tagger, which is the first tagger to use deep residual networks (ResNets). The tagger uses both word and character representations, and includes a novel residual bypass architecture. We evaluate the tagger separately on the semantic tagging task, on POS tagging, and in the multitask learning setting. In the multitask setting, the tagger significantly outperforms prior results on English Universal Dependencies POS tagging reaching 95.71% accuracy on UD v1.2 and 95.67% accuracy on UD v1.3.

*Chapter adapted from: **Bjerva, J.**, Plank, B., and Bos, J. (2016). Semantic tagging with deep residual networks. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3531–3541. The COLING 2016 Organizing Committee

4.1 Introduction

A key issue in computational semantics is the transferability of semantic information across languages. Many semantic parsing systems depend on sources of information such as POS tags (Pradhan et al., 2004; Copestake et al., 2005; Bos, 2008; Butler, 2010; Berant and Liang, 2014). However, these tags are often customised for the language at hand (Marcus et al., 1993) or massively abstracted, such as the Universal Dependencies tagset (Nivre et al., 2017, 2016a). Furthermore, POS tags are syntactically oriented, and therefore often contain both irrelevant and insufficient information for semantic analysis and deeper semantic processing. This means that, although POS tags are highly useful for many downstream tasks, they are unsuitable both for semantic parsing in general, and for tasks such as recognising textual entailment.

We present a novel set of semantic labels tailored for the purpose of multilingual semantic parsing. This tagset (i) abstracts over POS and named entity types; (ii) fills gaps in semantic modelling by adding new categories (for instance for phenomena like negation, modality, and quantification); and (iii) generalises over specific languages (see Section 4.2). We introduce and motivate this new task in this chapter, and refer to it as *semantic tagging*. Our experiments aim to answer the following two research questions, in order to answer **RQ 1**:

RQ 1a Can we use recent neural network architectures to implement a state-of-the-art neural sequence tagger, which can easily be expanded for multitask learning?

RQ 1b Semantic tagging is essential for deep semantic parsing, but can we find evidence that semtags are effective also for other NLP tasks, in a multitask learning setting?

We will first look at the semantic tag set, before going into the details of the neural network architecture used, and exploring these research questions.

4.2 Semantic Tagging

Semantic tagging, or *semtagging*, is the task of assigning semantic class categories to the smallest meaningful units in a sentence. In the context of this chapter these units can be morphemes, words, punctuation, or multi-word expressions. These tags are designed so as to facilitate semantic analysis and parsing in cross-linguistic settings, and to be as language neutral as possible, so as to be applicable to several languages (Abzianidze et al., 2017). The tag set is motivated by the observation that linguistic information traditionally obtained for deep processing is insufficient for fine-grained lexical semantic analysis. The widely used Penn Treebank (PTB) Part-of-Speech tagset (Marcus et al., 1993) does not make the necessary semantic distinctions, in addition to containing redundant information for semantic processing.

In particular, there are significant differences in meaning between the determiners *every* (universal quantification), *no* (negation), and *some* (existential quantification), but they all receive the `DT` (determiner) POS label in PTB. Since determiners form a closed class, one could enumerate all word forms for each class. Indeed some recent implementations of semantic parsing follow this strategy (Bos, 2008; Butler, 2010). This might work for a single language, but it falls short when considering a multilingual setting. Furthermore, determiners like *any* can have several interpretations and need to be disambiguated in context.

In addition to this, consider the following examples of redundant information of some POS tagsets, when considering semantic analysis. For instance, the tagset used in the PTB includes a distinction between `VBP` (present simple) and `VBZ` (present simple third person).

In the context of semantic analysis, this type of distinction is not necessary.

Semantic tagging does not only apply to determiners, but reaches all parts of speech. Other examples where semantic classes disambiguate are reflexive versus emphasising pronouns (both POS-tagged as PRP, personal pronoun); the comma, that could be a conjunction, disjunction, or apposition; intersective vs. subsective and privative adjectives (all POS-tagged as JJ, adjective); proximal vs. medial and distal demonstratives (see Example 2.1); subordinate vs. coordinate discourse relations; role nouns vs. entity nouns. ROL is used to separate roles from concepts, which is crucial in order to get accurate semantic behaviour (Abzianidze et al., 2017).

The set of semantic tags that we use in this chapter is established in a data-driven manner, considering four languages in a parallel corpus (English, German, Dutch and Italian). This first inventory of classes comprises 13 coarse-grained tags and 73 fine-grained tags (see Table 4.1).[2] As can be seen from this table and the examples given below, the tagset also includes named entity classes.

(4.1)

We	*must*	*draw*	*attention*	*to*	*the*	*distribution*	*of*	*this*
PRO	NEC	EXS	CON	REL	DEF	CON	AND	PRX

form	*in*	*those*	*dialects*	*.*
CON	REL	DST	CON	NIL

(4.2)

Ukraine	*'s*	*glory*	*has*	*not*	*yet*	*perished*	*,*	*neither*
GPE	HAS	CON	ENT	NOT	IST	EXT	NIL	NOT

her	*freedom*	*.*
HAS	CON	NIL

In Example 2.1,[3] both *this* and *those* would be tagged as DT. However,

[2]Experiments in this chapter are based on the tag inventory as detailed in this chapter. This is based on version 0.3 of the semantic tags used in the PMB in June 2016, and has since been revised.

[3]PMB 01/3421, Original source: Tatoeba

with our semantic tagset, they are disambiguated as PRX (proximal) and DST (distal). In Example 2.2,[4] *Ukraine* is tagged as GPE rather than NNP.

Annotated data

We use two semtag datasets. The Groningen Meaning Bank (GMB) corpus of English texts (1.4 million words) containing silver standard semantic tags obtained by running a simple rule-based semantic tagger (Bos et al., 2017). This tagger uses POS and named entity tags available in the GMB (automatically obtained with the C&C tools (Curran et al., 2007) and then manually corrected), as well as a set of manually crafted rules to output semantic tags. Some tags related to specific phenomena were hand-corrected in a second stage.

Our second dataset, the PMB, is smaller but equipped with gold standard semantic tags and used for testing (Abzianidze et al., 2017). It comprises a selection of 400 sentences of the English part of a parallel corpus. It has no overlap with the GMB corpus. For this dataset, we used the Elephant tokeniser, which performs word, multi-word and sentence segmentation (Evang et al., 2013). We then used the simple rule-based semantic tagger described above to get an initial set of tags. These tags were then corrected by a human annotator.

In order to enable the multitask learning setting, we look at the POS annotation in the English portion of the Universal Dependencies dataset, version 1.2 and 1.3 (Nivre et al., 2016a). An overview of the data used is shown in Table 4.2. We use the official training, development, and test splits on the UD data. For the semantic silver standard data set we split the data into 70% for training, 10% for development, and 20% for testing. We do not use any gold semantic tagging data for training or development, reserving the entire set for testing.

[4] PMB 05/0936, Original source: Tatoeba

Table 4.1: Semantic tags used in this chapter.

ANA	PRO	pronoun
	DEF	definite
	HAS	possessive
	REF	reflexive
	EMP	emphasizing
ACT	GRE	greeting
	ITJ	interjection
	HES	hesitation
	QUE	interrogative
ATT	QUA	quantity
	UOM	measurement
	IST	intersective
	REL	relation
	RLI	rel. inv. scope
	SST	subsective
	INT	intensifier
	SCO	score
LOG	ALT	alternative
	EXC	exclusive
	NIL	empty
	DIS	disjunct./exist.
	IMP	implication
	AND	conjunct./univ.
	BUT	contrast
COM	EQA	equative
	MOR	comparative pos.
	LES	comparative neg.
	TOP	pos. superlative
	BOT	neg. superlative
	ORD	ordinal
DEM	PRX	proximal
	MED	medial
	DST	distal
DIS	SUB	subordinate
	COO	coordinate
	APP	appositional
MOD	NOT	negation
	NEC	necessity
	POS	possibility
ENT	CON	concept
	ROL	role
NAM	GPE	geo-political ent.
	PER	person
	LOC	location
	ORG	organisation
	ART	artifact
	NAT	natural obj./phen.
	HAP	happening
	URL	url
EVE	EXS	untensed simple
	ENS	present simple
	EPS	past simple
	EFS	future simple
	EXG	untensed prog.
	ENG	present prog.
	EPG	past prog.
	EFG	future prog.
	EXT	untensed perfect
	ENT	present perfect
	EPT	past perfect
	EFT	future perfect
	ETG	perfect prog.
	ETV	perfect passive
	EXV	passive
TNS	NOW	present tense
	PST	past tense
	FUT	future tense
TIM	DOM	day of month
	YOC	year of century
	DOW	day of week
	MOY	month of year
	DEC	decade
	CLO	clocktime

Table 4.2: Overview of the semantic tagging data (ST Silver from the GMB, ST Gold from the PMB) and the Universal Dependencies data (UD), as of November 2016.

Corpus	Train (sents/toks)	Dev (sents/toks)	Test (sents/toks)	tags
ST Silver	42,599 / 930,201	6,084 / 131,337	12,168 / 263,516	66
ST Gold	n/a	n/a	356 / 1,718	66
UD	12,543 / 204,586	2,002 / 25,148	2,077 / 25,096	17

4.3 Method

To address **RQ 1a**, we will look at convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which are both highly prominent approaches in the recent natural language processing (NLP) literature (see Chapter 2). A recent development is the emergence of deep residual networks (ResNets), a building block for CNNs (see Section 2.6). In short, ResNets consist of several stacked residual units, which can be thought of as a collection of convolutional layers coupled with a shortcut which aids the propagation of the signal in a neural network. This allows for the construction of much deeper networks, since keeping a relatively clean information path in the network facilitates optimisation (He et al., 2016). ResNets have recently shown state-of-the-art performance for image classification tasks (He et al., 2015a, 2016), and have also seen some recent use in NLP (Östling, 2016; Conneau et al., 2016; Bjerva, 2016; Wu et al., 2016). However, no previous work has attempted to apply ResNets to NLP tagging tasks.

Our tagger is a hierarchical deep neural network consisting of a bidirectional Gated Recurrent Unit (GRU) network at the upper level,

and a CNN or a ResNet at the lower level, including an optional novel residual bypass function (cf. Figure 4.1). Bi-directional GRUs and LSTMs have been shown to yield high performance on several NLP tasks, such as POS tagging, named entity tagging, and chunking (Wang et al., 2015; Yang et al., 2016; Plank et al., 2016). We build on previous approaches by combining bi-GRUs with character representations from a basic CNN and ResNets.

4.3.1 Inception model

Due to the success of GoogLeNet on the ImageNet 2014 challenge, we experiment with a variant of their model codenamed *Inception* (Szegedy et al., 2015). The *Inception* model applies convolutions of different sizes (e.g. 1×1, 3×3, 5×5) in parallel to the output of the previous layer. In the system used by Szegedy et al. (2015), several such modules are stacked, forming a network of 22 layers. The intuition behind building a deep network with modules using varying convolutional patch sizes, is that the features learned by each layer are expected to be both more abstract and less spatially concentrated. Thus, in the lower layers of the model, the smaller patches are expected to capture most of the correlation statistics of the previous layer, whereas in deeper layers the larger patches are expected to do this (cf. Arora et al. (2014)). Adding a large amount of convolutions naïvely would explode the amount of parameters to be optimised. Hence, 1×1 convolutions are added before the expensive larger convolutions. This addition can be thought of as a dimensionality reduction and information compression step, analogous to the success of word embeddings (Szegedy et al., 2015).

We apply *Inception* modules directly following the layer containing embedded character representations, and pass this information through to the bi-GRU. Our main modification on the original *Inception* module is that we apply it in a sequence-to-sequence prediction task.

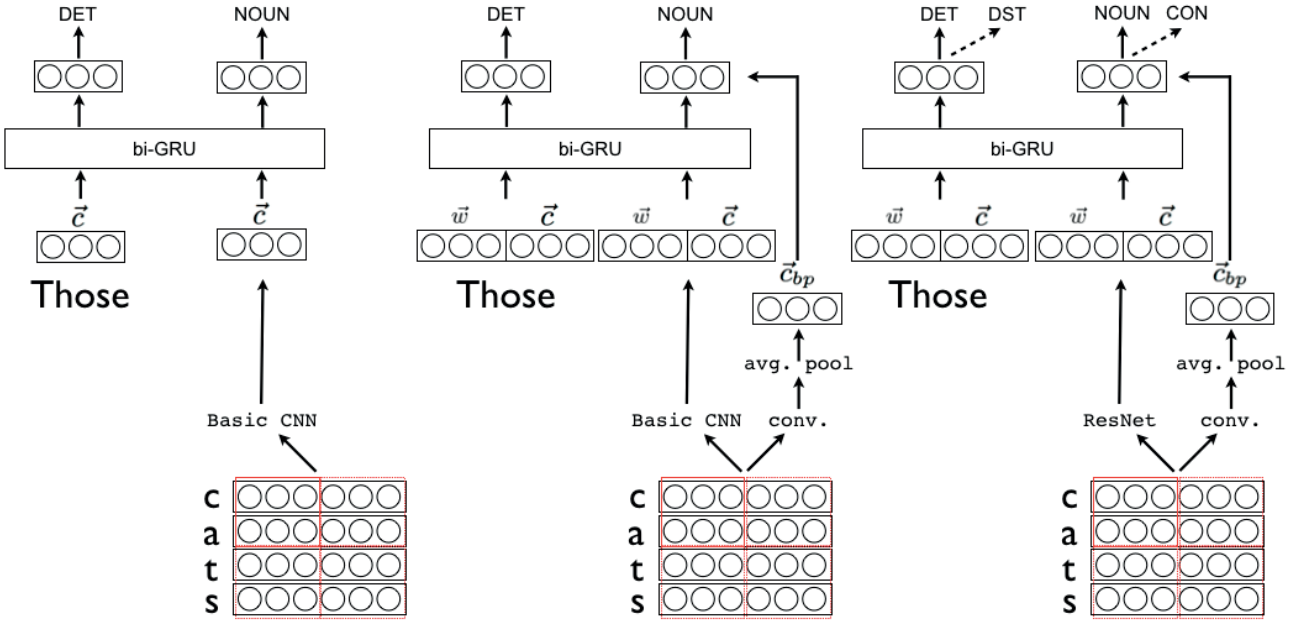


Figure 4.1: Model architecture. Left: Architecture with basic CNN char representations ($\vec{c}$), Middle: basic CNN with char and word representations and bypass ($\vec{c}_{bp} \wedge \vec{w}$), Right: ResNet with an auxiliary task and residual bypass ($+\text{AUX}_{bp}$).

4.3.2 Deep Residual Networks

We use Deep Residual Networks (ResNets), as introduced in Section 2.6. ResNets have recently been found to yield impressive performance in image recognition tasks, with networks as deep as 1001 layers (He et al., 2015a, 2016), and are thus an interesting and effective alternative to simply stacking layers. In this chapter we use the *assymetric* variant of ResNets similarly to what is described in Equation 9 in He et al. (2016), namely

$$z_{l+1} = z_l + F(\sigma(z_l)), \tag{4.3}$$

where z_l is the pre-activation of the lth layer, $\sigma(z_l) = \alpha_l$, F is a convolutional function. In other words, we add the pre-activation output of layer l to the output of a convolutional block over the same output, and set this to be the pre-activation of the following layer. This effectively gives the network a shortcut to the previous layer, which is useful when propagating errors backwards through the network.

In NLP, ResNets have been recently applied to morphological reinflection (Östling, 2016), native language identification (Bjerva et al., 2017; Kulmizev et al., 2017), sentiment analysis and text categorisation (Conneau et al., 2016), as well as machine translation (Wu et al., 2016). Our work is the first to apply ResNets to NLP sequence tagging tasks. We further contribute to the literature on ResNets by introducing a residual bypass function. The intuition is to combine both deep and shallow processing, which opens a path of easy signal propagation between lower and higher layers in the network.

4.3.3 Modelling character information and residual bypass

Using sub-token representations instead of, or in combination with, word-level representations has recently obtained a lot of attention due to their effectiveness (Sutskever et al., 2011; Chrupała, 2013; Zhang et al., 2015b; Chung et al., 2016; Gillick et al., 2015). There is much to

be said for approaching NLP using sub-token information. Doing so allows for much more compact models, since it is no longer necessary to learn weights associated with a large vocabulary, as when using word embeddings. Additionally, not relying on the linguistic artefact of white-space delimited text, opens up for learning whatever internal structure is the most appropriate for the task at hand (Gillick et al., 2015). Sub-token information can be seen as both a potential replacement for token information, or simply as a supplement, which is the approach we take in this chapter.

The use of sub-token representations can be approached in several ways. Plank et al. (2016) and Yang et al. (2016) use a hierarchical bi-directional RNN, first passing over characters in order to create word-level representations. Gillick et al. (2015) similarly apply an LSTM-based model using byte-level information directly. CNNs are used by dos Santos and Zadrozny (2014), who construct character-based word-level representations by running a CNN over the character representations of each word. All of these approaches have in common that the character-based representation is passed through the entire remainder of the network. Our work is the first to combine the use of character-level representations with both deep processing (i.e., passing this representation through the network) and shallow processing (i.e., bypassing the network in our residual bypass function). We achieve this by applying our novel residual bypass function to our character representations, inspired by the success of ResNets (depicted in Figure 4.1). In particular, we first apply the bypass to a CNN-based model achieving large gains over a plain CNN, and later evaluate its effectiveness in a ResNet. The bypass function allows both lower-level and higher-level features to be taken directly into account in the final layers of the network. The intuition behind using such a global residual function in NLP is that character information primarily ought to be of importance for the prediction of the current word. Hence, allowing these representations to bypass our bi-GRU

might be beneficial. This residual bypass function is not dependent on the usage of ResNets, and can be combined with other NN architectures as in our experiments. We define the penultimate layer, α_{n-1}, of a network with n layers, using a residual bypass, as

$$\alpha_{n-1} = \sigma(z_{n-1}) + \alpha_c, \tag{4.4}$$

where σ is some activation function, z_{n-1} is the pre-activation of layer $n-1$, and α_c is the activation of the character-level CNN, in the case of our experiments.

4.3.4 System description

The core of our architecture consists of a bi-GRU taking an input based on words and/or characters, with an optional residual bypass as defined in subsection 4.3.3. We experiment with a basic CNN, ResNets, a variant of the *Inception* model (Szegedy et al., 2015), and our novel residual bypass function. Our system is implemented in Keras using the Tensorflow backend (Chollet, 2015; Abadi et al., 2016).[5]

We represent each sentence using both a character-based representation (S_c) and a word-based representation (S_w). The character-based representation is a 3-dimensional matrix $\mathbb{S}_c^{s \times w \times d_c}$, where s is the zero-padded sentence length, w is the zero-padded word length, and d_c is the dimensionality of the character embeddings. The word-based representation is a 2-dimensional matrix $\mathbb{S}_w^{s \times d_w}$, where s is the zero-padded sentence length and d_w is the dimensionality of the word embeddings. We use the English Polyglot embeddings (Al-Rfou et al., 2013) in order to initialise the word embedding layer, but also experiment with randomly initialised word embeddings.

Word embeddings are passed directly into a two-layer bi-GRU (Chung et al., 2014). We also experimented using a bi-LSTM. However, we found GRUs to yield comparatively better validation data performance

[5]System code available at `https://github.com/bjerva/semantic-tagging`.

on semtags. We also observe better validation data performance when running two consecutive forward and backward passes before concatenating the GRU layers, rather than concatenating after each forward/backward pass as is commonplace in NLP literature.

We use CNNs for character-level modelling. Our basic CNN is inspired by dos Santos and Zadrozny (2014), who use character representations to produce local features around each character of a word, and combine these with a maximum pooling operation in order to create fixed-size character-level word embeddings. The convolutions used in this manner cover a few neighbouring letters at a time, as well as the entire character vector dimension (d_c). In contrast to dos Santos and Zadrozny (2014), we treat a word analogously to an image. That is to say, we see a word of n characters embedded in a space with dimensionality d_c as an image of dimensionality $n \times d_c$. This view gives us additional freedom in terms of sizes of convolutional patches used, which offers more computational flexibility than using only, e.g., $4 \times d_c$ convolutions. This view is applied to all CNN variations explored in this work.

To answer **RQ1b**, we investigate the effect of using semantic tags as an auxiliary task for POS tagging, in a multitask learning paradigm (see Chapter 3). Since POS tags are useful for many NLP tasks, it follows that semantic tags must be useful if they can improve POS tagging. A neural network is trained with respect to a given loss function, such as the cross-entropy between the predicted tag probability distribution and the target probability distribution. Recent work has shown that the addition of an auxiliary loss function can be beneficial to several tasks. For instance, Cheng et al. (2015) use this paradigm for language modelling, by predicting the next token while also predicting whether the sentence at hand contains a name. Plank et al. (2016) use the log frequency of the current token as an auxiliary task, and find this to improve POS tagging accuracy. Since our semantic tagging task is based on predicting fine semtags, which

can be mapped to coarse semtags, we add the prediction of these coarse semtags as an auxiliary task for the semtagging experiments. Similarly, we also experiment with POS tagging, where we use the fine semtags as an auxiliary information.

Hyperparameters

All hyperparameters are tuned with respect to loss on the semtag validation set. We use rectified linear units (ReLUs) for all activation functions (Nair and Hinton, 2010), and apply dropout with $p = 0.1$ to both input weights and recurrent weights in the bi-GRU (Srivastava et al., 2014). In the CNNs, we apply batch normalisation (Ioffe and Szegedy, 2015) followed by dropout with $p = 0.5$ after each layer. In our basic CNN, we apply a 4×8 convolution, followed by 2×2 maximum pooling, followed by 4×4 convolution and another 2×2 maximum pooling. Our ResNet has the same setup, with the addition of a residual connection. We also experimented with using average pooling instead of maximum pooling, but this yielded lower validation data performance on the semantic tagging task. We set both d_c and d_w to 64. All GRU layers have 100 hidden units. All experiments were run with early stopping monitoring validation set loss, using a maximum of 50 epochs. We use a batch size of 500. Optimisation is done using the ADAM algorithm (Kingma and Ba, 2014), with the categorical cross-entropy loss function as training objective. The main and auxiliary loss functions have a weighting parameter, λ. In our experiments, we weight the auxiliary task with $\lambda = 0.1$, as set on the semtag auxiliary task, and a weighting of $\lambda = 1.0$ for the main task.

Multi-word expressions (MWEs) are especially prominent in the semtag data, where they are annotated as single tokens. Pre-trained word embeddings are unlikely to include entries such as 'International Organization for Migration', so we apply a simple heuristic in order to avoid treating most MWEs as unknown words. That is to say, the representation of a MWE is set to the sum of the individual

embeddings of each constituent word, such that

$$\overrightarrow{\text{mwe}} = \sum_{w \in \text{mwe}} \vec{w}, \tag{4.5}$$

where mwe is the MWE at hand, $w \in mwe$ is every word in this mwe, and $\vec{w}$ is the embedded vector representation of that word.

4.4 Evaluation

We evaluate our tagger on two tasks: semantic tagging and POS tagging. Note that the tagger is developed solely on the semantic tagging task, using the GMB silver training and validation data. Hence, no further fine-tuning of hyperparameters for the POS tagging task is performed. We calculate significance using bootstrap resampling (Efron and Tibshirani, 1994). The following independent variables are manipulated in our experiments:

1. character and word representations ($\vec{w}, \vec{c}$);
2. residual bypass for character representations ($\vec{c}_{bp}$);
3. convolutional representations (Basic CNN and ResNets);
4. auxiliary tasks (using coarse semtags on ST and fine semtags on UD).

We compare our results to four baselines:

1. the most frequent baseline per word (MFC), where we assign the most frequent tag for a word in the training data to that word in the test data, and unseen words get the global majority tag;
2. the trigram statistic based TNT tagger which offers a slightly tougher baseline (Brants, 2000);

3. the BI-LSTM baseline, running the off-the-shelf state-of-the-art POS tagger for the UD dataset (Plank et al., 2016) (using default parameters with pre-trained Polyglot embeddings);

4. we also use a baseline consisting of running our own system with only a BI-GRU using word representations ($\vec{w}$), with pre-trained Polyglot embeddings.

4.4.1 Experiments on semantic tagging

We evaluate our system on two semantic tagging (ST) datasets: our silver semtag dataset and our gold semtag dataset. For the +AUX condition we use coarse semtags as an auxiliary task. Results from these experiments are shown in Table 4.3.

4.4.2 Experiments on Part-of-Speech tagging

We evaluate our system on v1.2 and v1.3 of the English part of the Universal Dependencies (UD) data. We report results for POS tagging alone, comparing to commonly used baselines and prior work using LSTMs, as well as using the fine-grained semantic tags as auxiliary information. For the +AUX condition, we train a single joint model using a multi-task objective, with POS and ST as our two tasks. This model is trained on the concatenation of the ST silver data with the UD data, updating the loss of the respective task of an instance in each iteration. Hence, the weights leading to the UD output layer are not updated on the ST silver portion of the data, and vice-versa for the ST output layer on the UD portion of the data. Results from these experiments are shown in Table 4.3.

4.4.3 The Inception architecture

For comparison with the ResNet, we evaluate the *Inception* architecture on ST Silver data, and UD v1.2 and v1.3 (see Table 4.4).

Table 4.3: Experiment results on semtag (ST) and Universal Dependencies (UD) test sets (% accuracy). MFC indicates the per-word most frequent class baseline, TNT indicates the TNT tagger, and BI-LSTM indicates the system by Plank et al. (2016). BI-GRU indicates the $\vec{w}$ only baseline. $\vec{w}$ indicates usage of word representations, $\vec{c}$ indicates usage of character representations, and $\vec{c}_{bp}$ indicates usage of residual bypass of character representations. The +AUX column indicates the usage of an auxiliary task.

		ST Silver	ST Gold	UD v1.2	UD v1.3
BASELINES	MFC	84.64	77.39	85.06	85.07
	TNT	92.09	80.73	92.66	92.69
	BI-LSTM	94.98	82.96	95.17	95.04
	BI-GRU	94.26	80.26	94.39	94.32
BASIC CNN	$\vec{c}$	91.39	69.21	77.63	77.51
	$\vec{c}_{bp}$	90.18	65.77	83.53	82.89
	$\vec{c}_{bp} \wedge \vec{w}$	94.63	76.83	94.68	94.89
	$+\text{AUX}_{bp}$	94.53	80.73	95.19	95.34
RESNET	$\vec{c}$	94.39	76.89	92.65	92.63
	$\vec{c} \wedge \vec{w}$	95.14	**83.64**	94.92	94.88
	+AUX	94.23	74.84	**95.71**	**95.67**
	$\vec{c}_{bp}$	94.23	75.84	92.45	92.86
	$\vec{c}_{bp} \wedge \vec{w}$	**95.15**	82.18	94.73	94.69
	$+\text{AUX}_{bp}$	94.58	73.73	95.51	95.57

Table 4.4: Results when using the *Inception* architecture on ST and UD data. $\vec{w}$ indicates usage of word representations, $\vec{c}$ indicates usage of character representations, and $\vec{c}_{bp}$ indicates usage of residual bypass of character representations.

	$\vec{c}$	$\vec{c}_{bp}$	$\vec{c}_{bp} \wedge \vec{w}$	ResNet
ST Silver	94.40	93.32	94.64	**95.15**
UD v1.2	90.82	89.78	**95.07**	94.73
UD v1.3	91.12	89.55	**94.90**	94.69

4.4.4 Effect of pre-trained embeddings

In our main experiments, we initialise the word embedding layer with pre-trained polyglot embeddings. We compare this with randomly initialising this layer from a uniform distribution over the interval $[-0.05, 0.05)$, without any pre-training. Results from these experiments are shown in Table 4.5.

Table 4.5: Results under the $\vec{c}_{bp} \wedge \vec{w}$ and $\vec{c}_{bp} \wedge \vec{w}$+AUX conditions, on ST and UD data, using randomly initialised word embeddings. Change in accuracy is indicated in brackets.

	$\vec{c}_{bp} \wedge \vec{w}$	+AUX
ST Silver	95.11 (-0.04)	94.57 (-0.01)
UD v1.2	91.94 (-2.79)	94.90 (-0.61)
UD v1.3	92.00 (-2.69)	94.96 (-0.61)

4.5 Discussion

4.5.1 Performance on semantic tagging

The overall best system is the ResNet combining both word and character representations $\vec{c} \wedge \vec{w}$. It outperforms all baselines, including the recently proposed RNN-based bi-LSTM. On the ST silver data, a significant difference ($p < 0.01$) is found when comparing our best system to the strongest baseline (BI-LSTM). On the ST gold data, we observe significant differences at the alpha values recommended by Søgaard et al. (2014), with $p < 0.0025$. The residual bypass effectively helps improve the performance of the basic CNN. However, the tagging accuracy of the CNN falls below baselines. In addition, the large gap between gold and silver data for the CNN shows that the CNN model is more prone to overfitting, thus favouring the use of the ResNet.

Adding the coarse-grained semtags as an auxiliary task only helps for the weaker CNN model. The ResNet does not benefit from this additional information, which is already captured in the fine-grained labels.

It is especially noteworthy that the ResNet character-only system performs remarkably well, as it outperforms the BI-GRU and TNT baselines, and is considerably better than the basic CNN. Since performance increases further when adding in $\vec{w}$, it is clear that the character and word representations are complimentary in nature. The high results for characters only are particularly promising for multilingual language processing, as such representations allow for much more compact models (see, e.g., Gillick et al. (2015)). This further indicates that ResNet-based character representations can almost account for the same amount of compositionality as word representations.

4.5.2 Performance on Part-of-Speech tagging

Our system was tuned solely on semtag data. This is reflected in, e.g., the fact that even though our $\vec{c} \wedge \vec{w}$ ResNet system outperforms the Plank et al. (2016) system on semtags, we are substantially outperformed on UD 1.2 and 1.3 in this setup. However, adding an auxiliary task based on our semtags markedly increases performance on POS tagging. In this setting, our tagger outperforms the BI-LSTM system, and results in new state-of-the-art results on both UD 1.2 (95.71% accuracy) and 1.3 (95.67% accuracy). The difference between the BI-LSTM system and our best system is significant at $p < 0.0025$.

The fact that the semantic tags improve POS tagging performance reflects two properties of semantic tags. Firstly, it indicates that the semantic tags carry important information which aids the prediction of POS tags. This should come as no surprise, considering the fact that the semtags abstract over and carry more information than POS tags. Secondly, it indicates that the new semantic tagset and released

dataset are useful for downstream NLP tasks. This could be done indirectly, by running a POS tagger which has been trained in a multitask setting with semtags. Alternatively, the semtags would likely be useful features for downstream tasks. In this chapter we show this by using semtags as an auxiliary task. In future work we aim to investigate the effect of introducing the semtags directly as features into the embedded input representation.

4.5.3 Inception

The experiments using the *Inception* architecture showed that the ResNet architecture we used performs better on semantic tagging. The first of these results is in line with results in, e.g., image recognition, where ResNets are also superior to *Inception* (He et al., 2015a, 2016). On UD PoS tagging, however, results using *Inception* were marginally better. This might be explained by the fact that the ResNet architecture was rather heavily tuned on semantic tagging, whereas *Inception* was tuned to a lesser extent. Nonetheless, both architectures outperform the use of a standard relatively shallow CNN, indicating that they may indeed be more suitable for similar tasks, given similar amounts of data.

4.5.4 Residual bypass

Our novel residual bypass function outperforms corresponding models without residual bypass in some cases. Notably for POS tagging with a standard CNN, the increase in tagging accuracy is around 5%. Combining this with a ResNet does not have a large effect on tagging performance, resulting in slightly higher accuracy on UD 1.3, but lower on UD 1.2 and semtags. This indicates that, although using a residual bypass allows for the character information to propagate more easily, this is not crucial for the model to capture subtoken information and use this effectively. An interesting possibility for

future research is to investigate the use of a residual bypass on word-level representations.

4.5.5 Pre-trained embeddings

For semantic tagging, the difference with random initialisation is negligible, with pre-trained embeddings yielding an increase in about 0.04% accuracy. For POS tagging, however, using pre-trained embeddings increased accuracy by almost 3 percentage points for the ResNet.

4.6 Conclusions

In this chapter, we first introduced a semantic tagset tailored for multilingual semantic parsing. We compared tagging performance using standard CNNs and the recently emerged ResNets. For semantic tagging, ResNets are more robust and result in our best model. Combining word and ResNet-based character representations helps to outperform state-of-the-art taggers on semantic tagging, while allowing for straightforward extension to an MTL paradigm (**RQ 1a**). Since we were interested in seeing whether the new tagset could be informative for other tasks, we used semantic tagging as an auxiliary task for PoS tagging. This yielded state-of-the-art performance on the English UD 1.2 and 1.3 POS datasets, showing that semantic tags are informative for other NLP tasks (**RQ 1b**). The fact that using semantic tags aided POS tagging raises the question of in which cases it is useful to have an auxiliary tagging task (**RQ 2**), which is explored in the following chapter.