One Model to Rule them All

Bjerva, Johannes

CHAPTER 3

Multitask Learning and Multilingual Learning

Abstract | In this chapter, we build upon the background knowledge of neural networks presented in the previous chapter. We will focus on different, but highly related, paradigms – multitask learning, and multilingual model transfer. Multitask learning is first presented in a general context, and then in the context of neural networks, which is the primary focus of this thesis. We will then look at multilingual approaches in NLP, again first in a general context, and then in the context of model transfer with multilingual word representations, which is the secondary focus of this thesis. In this thesis, we consider the first setting in Part II, the second setting in Part III, and include an outlook for a combined multilingual/multitask paradigm in Part IV.

3.1 Multitask Learning

In Natural Language Processing (NLP), and machine learning (ML) in general, the focus is generally on solving a single task at a time. For instance, one might invest significant amounts of time in making a Part-of-Speech tagger or a parser. However, fact is that many tasks are related to one another. The aim of multitask learning (MTL) is to take advantage of this fact, by attempting to solve several tasks simultaneously, while taking advantage of the overlapping information in the training signals of related tasks (Caruana, 1993, 1997). When the tasks are related to each other, this approach can improve generalisation, partially since it provides a source of inductive bias, and since it allows for leveraging larger amounts of more diverse data.[1] Additionally, since related tasks can often make use of similar representations, this can lead to the tasks being learnt even better than when training on a single task in isolation.

The use of MTL is skyrocketing in NLP, and has been applied successfully to a wide range of tasks, for instance sequence labelling such as POS tagging (Collobert and Weston, 2008; Plank et al., 2016), semantic tagging (Bjerva et al., 2016b), as well as chunking and supertagging (Søgaard and Goldberg, 2016). In addition to this, it is the primary focus of this thesis, and having some background knowledge on this will be useful for the following chapters. The first part of this chapter is an attempt at providing an understanding of what MTL is and how it is applied. While some general MTL scenarios are covered, the focus will be on MTL in the context of neural networks, and in the context of NLP.

[1] Generally speaking, it is beneficial to have access to more data when training an ML model.

3.1.1 Non-neural Multitask Learning

Before going into MTL in neural networks (NNs), we first take a look at the usage of this paradigm in other frameworks. Generally speaking, we seek to exploit the fact that there are many tasks which are somehow related to one another (Caruana, 1993, 1997; Thrun and Pratt, 1998). For instance, MTL can have the role of being a distant supervision signal, in the sense that the tasks used might be fairly distantly related. Additionally, since MTL plays the role as a regulariser (see Chapter 2), and lowers the risk of overfitting (Baxter, 1997; Baxter et al., 2000), MTL often improves generalisation. This is in part because MTL reduces *Rademacher complexity* (Baxter et al., 2000; Maurer, 2006).[2] Furthermore, MTL will push the weights of a model towards representations which are useful for more than one task. Finally, MTL can be seen as a method of dataset augmentation, as it allows for using more data than when only considering a single task at a time.

A commonly made assumption in MTL is that only a handful of parameters or weights (see Chapter 2) ought to be shared between tasks, and conversely that most parameters should not be shared (Argyriou et al., 2007). This can intuitively be understood by considering that only a few features useful for a task t_1 might be useful for another task t_2. For instance, imagine that we are building a joint POS tagger and language identification system. A feature capturing capitalised words preceded by a determiner will both be a decent indicator of the language being, e.g., German, as well as that the capitalised word is a noun. Other features, on the other hand, such as one indicating that the language is likely to be Norwegian or Danish if the letter ø is encountered, is not likely to be beneficial for POS tagging at all. In other words, this type of *parameter sparsity* can be phrased as that

[2] A lower Rademacher complexity essentially indicates that a class of functions is easier to learn.

most parameters should not be shared, as many parameters are task specific. In this type of approach, all shared parameters are generally considered by all tasks involved. This puts the system at a relatively large risk of negative transfer, if one tries to combine this approach with tasks which are only slightly related. In NLP we are often interested in exploiting even relatively weak training signals, which makes this particularly problematic.

Another approach is to learn clusters of tasks, which allows for letting related tasks share certain parameters, and relatively unrelated tasks perhaps only a few. Such approaches have in common that they assume that the parameters which are beneficial for each other are geometrically close to one another in n-dimensional space (Evgeniou and Pontil, 2004; Kang et al., 2011). Other work has come up with other definitions of task similarities. For instance, Thrun and O'Sullivan (1995) consider two tasks to be similar simply if one improves performance on the other. While other approaches to MTL have been used in the past, such as Daumé III (2009) who approach MTL from a Bayesian perspective, and Toutanova et al. (2005) who train a joint classifier for semantic role labelling with automatically generated auxiliary tasks, the perhaps most popular approach in NLP is parameter sharing in NNs.

3.1.2 Neural Multitask Learning

We now turn to the main method used in this thesis, namely neural MTL. There are two main approaches to this, differing in the manner in which parameters are shared – *hard* and *soft* parameter sharing. Currently, the less popular variant of the two in NLP is soft parameter sharing, and will not be covered in detail. Briefly put, in this setting, parameters are constrained in a similar manner to the *parameter sparsity* approach. That is to say, the parameters between tasks are encouraged to be similar to one another, which allows for some transfer between tasks, or between languages (Duong et al.,

2015). However, as parameters are not explicitly shared between tasks, the risks of negative transfer are relatively low in this setting. This approach is not explored in this thesis as hard parameter sharing offers several advantages, including ease of implementation, and computational effectivity, as the amount of parameters is kept almost constant as compared to having a single task.

Hard parameter sharing is currently more common, perhaps mainly due to the ease with which a neural MTL system with several tasks can be created. This is the type of MTL discussed in the seminal works by Caruana (1993, 1997). In this thesis we consider research questions tied to this type of MTL in the context of NLP, partially due to the versatility of the paradigm. Apart from allowing for considering data from several tasks simultaneously, even corpora in different languages might be used in this approach, given some sort of unified input representations.[3] Then, if the output labels between tasks correlate with one another to some extent, it seems quite intuitive that this approach should be beneficial.

In NLP, MTL is generally approached from the perspective that there is some *main* task, i.e., the task in which we are interested, and some *auxiliary* task, which should improve the main task. It is important to note, however, that these labels are quite arbitrary.[4] There is not necessarily anything to distinguish a main task from an auxiliary task in an NN. One might lower the weighting of the auxiliary task (i.e. multiply the loss for each batch by some $\lambda < 1$), but this strategy appears to be relatively rare in the literature.

A common way of implementing hard parameter sharing, is to have a stack of layers for which weights are updated with respect to all tasks, with at least two output layers, each with task-specific weights (see Figure 3.1). Concretely, consider that we have t corpora

[3]This is covered further in the second half of this chapter.

[4]The exception being cases in which the performance on the auxiliary task is disregarded in favour of the main task performance.

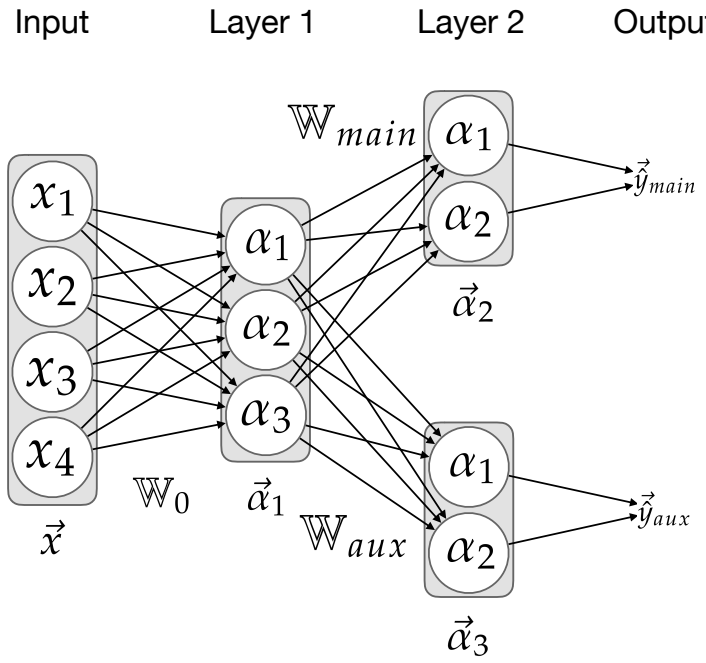

Figure 3.1: Common MTL architecture (bias units omitted for clarity).

with different annotations, each containing pairs of input and output sequences $(\vec{x}, \vec{y}^t)$ for a single task. While the inputs x will largely be part of the same vocabulary, and can be shared across tasks, the tag sets used, and therefore the labels $(y^0 \dots y^t)$ differ. Note that the vocabularies in the tasks at hand do not necessarily need to overlap, but when considering a single natural language, this tends to be the case. A common approach when training is to randomly sample such sequence pairs, predict a label distribution $\vec{\hat{y}}^t$, and update model parameters as calculated by the loss relative to the true label distribution $\vec{y}^t$ with backpropagation (see Chapter 2 for an overview

of this). Each task t has a task-specific classifier ($f_{t=main}, f_{t=aux}$) with its own weight matrix ($\mathbb{W}_{main}, \mathbb{W}_{aux}$). The output of the task-specific layer is then calculated using the softmax function (cf. Section 2.3.2), such that

$$\vec{\hat{y}}_t = \text{softmax}(\mathbb{W}_t \vec{\alpha}_n + b), \tag{3.1}$$

where $\vec{\alpha}_n$ denotes the activations of the layer before the output layer.

This architecture is common in NLP, with weights typically shared between the *main* and *auxiliary* task at all layers, up to the task-specific classification layer (i.e. the output layer). A multitude of other possibilities do exist, as the task-specific output layers can be attached anywhere in the network. This can be advantageous, as Søgaard and Goldberg (2016) found that including the lower-level task supervision at lower levels in the network was useful, in the case of using the low-level task of POS tagging in combination with CCG supertagging (i.e. assigning CCG lexical categories). Most related work, including the experiments in this thesis, apply multitask learning akin to what is shown in Figure 3.1.

Neural Multitask Learning in Natural Language Processing

Hard parameter sharing in NNs is the target of considerable attention in the recent NLP literature. Practically speaking, there appear to be two main approaches to MTL in the NLP literature. Some work, such as Ando and Zhang (2005), Collobert and Weston (2008), Søgaard and Goldberg (2016), Plank et al. (2016), Bjerva et al. (2016b), and Augenstein and Søgaard (2017) take the approach of exploiting seemingly related NLP tasks, based on some linguistic annotation. Other work, e.g., Plank (2016), and Klerke et al. (2016), take the approach of exploiting data from non-linguistic sources (keystroke data and eye gaze data, respectively). While these approaches are both useful and interesting, the focus of this thesis is the first approach,

specifically in which an NLP sequence prediction task is used as an auxiliary task for some other NLP sequence prediction task. This is partially motivated by the fact that using a word-level input for all tasks, allows for a one-to-one mapping between labels in different tag sets (given a specific token in context), which in turn opens up for the information-theoretic approach considered in Chapter 5.

3.1.3 Effectivity of Multitask Learning

Plenty of studies demonstrate the success of MTL, such as in computer vision (Torralba et al., 2007; Loeff and Farhadi, 2008; Quattoni et al., 2008), genomics (Obozinski et al., 2010), and the aforementioned NLP studies. Apart from relatively straight-forward results showing that MTL is often beneficial, efforts have been put into experimentally investigating when and why MTL is advantageous in NLP. Martínez Alonso and Plank (2017) look at a collection of semantic main tasks while using morphosyntactic and frequency-based tasks as auxiliary tasks. They find that the success of an auxiliary tasks depends on the distribution of the auxiliary task labels, e.g., the distribution's entropy and kurtosis.[5] Bingel and Søgaard (2017) present a large systematic study of MTL in a collection of NLP tasks. They find that certain dataset characteristics are predictors of auxiliary task effectivity, corroborating the findings of Martínez Alonso and Plank (2017), and also show that MTL can help target tasks out of local minima in the optimisation process. In Bjerva (2017b), it is argued that entropy is not sufficient for explaining auxiliary task effectivity, and that measures which take the joint distribution between tasks into account offer more explanatory value (this is elaborated in Chapter 5).

In terms of data sizes Benton et al. (2017) suggest that MTL is effective given limited training data for the main task. Luong et al. (2015), however, highlight that the auxiliary task data should not out-

[5]The kurtosis of a distribution is essentially a measure of its tailedness.

size the main task data – this is contradicted by Augenstein and Søgaard (2017), who highlight the usefulness of an auxiliary task when abundant data is available for such a task, and little for the main task. Finally, Mou et al. (2016) investigate transferability of neural network parameters, by attempting to initialise a network for a main task with weights which are pre-trained on an auxiliary task, and highlight the importance of similarities between tasks in such a setting. Finally, a promising recent innovation is that of sluice networks, in which a NN learns which parts of hidden layers to share between tasks, and to what extent (Ruder et al., 2017).

3.1.4 When MTL fails

The cases in which MTL does not work are also deserving of attention. While up until now we have assumed that applying MTL is a piece of cake, there are times when one adds an auxiliary task, causing the system to collapse like a house of cards. This type of performance loss is referred to as *negative transfer*, and can occur when two unrelated tasks share parameters. This is generally something to avoid, as there are few, if any, advantages to worsening the generalisation ability of the network. However, such results are rarely shared in the community, in part due to the *file drawer problem* (Rosenthal, 1979). In short, the problem is that it is impossible to access, or even know of, studies which have been conducted and not published. In the case of MTL, this issue might be alleviated by publishing results on all auxiliary tasks experimented with, even if only one or two such tasks improved performance.

3.2 Multilingual Learning

In the second half of this chapter, we turn to multilingual approaches. Many languages are similar to each other in some respect, and similarly to related tasks, this fact can also be exploited in order to im-

prove model performance with respect to, e.g., a given language. While there are many approaches to multilingual NLP, with various use cases, the focus in this thesis is on model transfer, in which a single model is shared between languages. We will nonetheless begin with an overview of the most common approaches.

As an example, consider NLP tagging tasks, which can be summed up as learning to assign a sequence of tags from a tag set t to a sequence of tokens in language l. In cross-lingual multitask NLP settings, there are many l/t pairs which do not have any annotated data. For instance, there is (at the time of writing) no annotated data for Welsh in the Universal Dependencies (Nivre et al., 2017). However, many NLP systems require input data from specific tag sets. For instance, the Stanford Neural Network Dependency parser requires POS tags in its input (Chen and Manning, 2014), whereas the semantic parser Boxer requires semantic tags in its input (Bos, 2008; Abzianidze et al., 2017). Hence, for such tools to be applicable in multilingual settings, the tags they rely on need to be available for other languages as well, which highlights the importance of approaches which deal with this. There are three frequently used approaches to solving this problem:

1. human annotation;
2. annotation projection;
3. model transfer.

Although serious efforts have gone into furthering these approaches, they all have considerable drawbacks. In brief, human annotation is time consuming and expensive, annotation projection is only applicable to texts which are both translated and aligned, and model transfer is generally only used in mono-lingual or mono-task settings.

3.2.1 Human Annotation

Generally speaking, annotating data manually is a very expensive and time-consuming manner of, e.g., producing some sort of linguistic labels for a sentence. Although the process can be alleviated with gamification (Venhuizen et al., 2013; Chamberlain, 2014; Jurgens and Navigli, 2014; Bos and Nissim, 2015), considerable time and effort still needs to be invested into creating such crowd-sourcing systems.

3.2.2 Annotation Projection

Given an annotated sentence in a source language and a translation of that sentence in a target language, it is possible to transfer, or project, the annotation from the source language to the target language. This approach is known as annotation projection, and relies on having access to parallel text for which at least one source language is annotated (Yarowsky et al., 2001; Hwa et al., 2005). Usually, word alignments are used in order to project linguistic labels from source to target. The resulting annotations can then be used to train a new monolingual system for the target language(s). This approach has been applied successfully to various tasks, primarily syntactic parsing (Hwa et al., 2005; Tiedemann, 2014; Rasooli and Collins, 2015; Agić et al., 2016), POS tagging (Yarowsky et al., 2001), and recently also semantic parsing (Evang and Bos, 2016; Evang, 2016).

Annotation projection has two main drawbacks. Primarily, it is only applicable to texts which are both translated and aligned, whereas the majority of available texts are monolingual. Furthermore, this approach relies heavily on the quality of the automatic word alignments. Word-aligning parallel text is not always successful, for instance with very dissimilar languages, insufficient statistics, or bad translations (Östling, 2014, 2015). Another approach for annotation projection relies on automatic translation. This works by applying a machine translation (MT) system to generate a parallel text for which

source language annotation exists (Tiedemann et al., 2014). In other words, in addition to the difficulties of the annotation projection approach, this method places high requirements on availability of parallel texts for training an MT model. In addition to these prerequisites, the involvement of a fully-fledged MT system in an annotation pipeline, will in itself increase its complexity severely.

3.2.3 Model Transfer

Model transfer deals with learning a single model which is shared between several languages (Zeman and Resnik, 2008; McDonald et al., 2011a).[6] This type of approach has been explored extensively in previous work. Multilingual model transfer has been successfully applied to, e.g., POS tagging (Täckström et al., 2013), and syntactic parsing (Täckström, 2013; Ammar et al., 2016). This is commonly done by using delexicalised input representations, as in the case of parsing (Zeman and Resnik, 2008; McDonald et al., 2011a; Täckström et al., 2012, 2013). A related situation, is the case of exploiting language similarities in order to train models for low-resource languages (see, e.g., Georgi et al. (2010)).

In this thesis, model transfer is framed as a special case of MTL. That is to say, each language in the model can be seen analogously to a task. This means that we are also free to choose whether we want to code tag predictions jointly as a single output layer, or have one separate output layer per language. As with MTL with multiple tasks, we consider the same specific type of MTL across languages, namely hard parameter sharing in neural networks.

A common approach in parsing is to delexicalise the input representations in order to enforce uniformity across languages, by training a parser on sequences of PoS tags rather than sequences of words (Zeman and Resnik, 2008; McDonald et al., 2011a). However, as we

[6] Note the similarities to multitask learning with hard parameter sharing.

are looking at predicting such tags, we approach this by using input representations which are shared across languages. This allows for training a neural network for several languages simultaneously in a language-agnostic manner, while still taking lexical semantics into account. Apart from this advantage, implementing a system in this manner is straightforward. Additionally, this approach offers the possibility of out-of-the-box zero-shot learning, as simply adding input representations for a different language is sufficient to enable this.

Zero-shot learning is the problem of learning to predict labels y which have not been seen during training. This is especially relevant in cases such as MT, in which, e.g., many of the target forms which need to be produced for languages with rich morphology have not been seen. In recent years, zero-shot learning has become increasingly popular, for instance in image recognition (Palatucci et al., 2009; Socher et al., 2013b). Recently, it has also been applied to MT, resulting in a model which even allows for translation into unseen languages (Johnson et al., 2016). One way of enabling zero-shot learning, is to use shared input representations. For instance, in the case of character-based models, we can simply use the same alphabet in the inputs and outputs of each system, as in Östling and Bjerva (2017) and Bjerva (2017a). In the case of word level input representations, one can employ word embeddings living in the same space, regardless of language.

3.2.4 Model Transfer with Multilingual Input Representations

Looking further at the problem of model transfer across languages, consider the following example, of an English sentence and its translation, as two separate input sequences to a neural network, with their corresponding annotated output sequences.[7]

[7] PMB 01/3421. Original source: Tatoeba.

(3.2) *We must draw attention to the distribution of*
PRON VERB VERB NOUN ADP DET NOUN ADP
this form in those dialects .
DET NOUN ADP DET NOUN PUNCT

(3.3) *Wir müssen die Verbreitung dieser Form in diesen*
PRON VERB DET NOUN DET NOUN ADP DET
Dialekten beachten .
NOUN VERB PUNCT

Although the surface forms of these two sentences differ, as one is in English and one in German, multilingual word representations for the corresponding words in these two sentences ought to be close to one another. Hence, if the NN only sees the English sentence in training, and the German sentence during test time, it ought to be fairly successful in tagging this 'unseen' sentence with suitable tags.[8] However, one question is whether having access to the same sentence in a typologically more distance language such as Japanese also would be useful (this is approached in Chapter 5).

In order for such an approach to work, it is necessary that words with similar meanings in different languages are represented in a fairly similar way. How do we arrive at word representations with such properties? In the next few sections we will look at this, beginning at simple monolingual representations, and leading up to bilingual and multilingual representations.

[8]Considering that the semantic content of the two sentences ought to be highly similar, one could regard the translated sentence to be 'seen' if the original sentence was in the training data. This has the further implication that one, in this type of experiments, must take care not to allow corresponding sentences to occur in both training and evaluation data.

3.2.5 Continuous Space Word Representations

In many NLP problems, we are concerned with processing some word-like unit, in order to arrive at some linguistically motivated and appropriate label. In Section 2.3.1, we considered bag-of-words models for tasks such as this. To recap, in this type of model we assign an index to each unique word. Each word is then represented by a vector $\vec{x}$, with a dimensionality equal to the size of the vocabulary, since each word requires its own index. As an example, consider a vocabulary size of five words, with three of those words being *cat*, *dog*, and *coastal*, with their corresponding vector representations, such that

$$\begin{aligned} \vec{x}_{cat} &= [0, 0, 1, 0, 0], \\ \vec{x}_{dog} &= [0, 0, 0, 0, 1], \\ \vec{x}_{coastal} &= [0, 1, 0, 0, 0]. \end{aligned} \tag{3.4}$$

In NLP, we are often interested in comparing words with one another, either simply in order to have some measure of their similarity, or because we are interested in the fact that similar words tend to have similar properties in down-stream tasks. For instance, the words *cat* and *dog* are likely to have the same or similar linguistic analyses in many cases, such as both being tagged with the PoS tag `NOUN`. A commonly used similarity measure between vectors is the cosine distance. In this setting, a word representation as presented above is somewhat problematic, as the distances between the three words are equal, although we want a higher similarity between *cat* and *dog*.

The representation we have seen so far is known as a *sparse* feature representation, as each word is represented by a vector of zeroes with one element set to one, also known as a *one-hot vector*. Apart from the drawback of similarity, this type of input representation can run into other problems, such as the dimensionality of the representations becoming too large to handle as vocabulary size grows. This can be remedied in many ways, for instance by applying

dimensionality reduction algorithms. Commonly used algorithms include singular value decomposition (SVD), and random indexing (Kanerva et al., 2000; Sahlgren, 2005). This does not help with the problem of similarities, however.

It turns out that one can arrive at word representations with nice properties of similarity by taking advantage of the *distributional hypothesis*:

> *'Semantics is partly a function of the statistical distribution of words.'*
> –Harris (1954)

> *'You shall know a word by the company it keeps.'*
> –Firth (1957, p.11)

This means that the semantic content of a given word is related to other words occurring in similar contexts. Furthermore, Harris (1954) claims that the strength of this relation is proportional to the similarity between two words, such that if two words w_1 and w_2 are more similar in meaning than w_1 and w_3, then the relative distribution of the first pair will be more similar than that of the second pair. One way of implementing this type of distributional semantics is to count word co-occurrences in a large corpus. Let us now assign the two remaining indices in the five-dimensional representation used above to the words *pet* and *water*, such that

$$\begin{aligned} \vec{x}_{pet} &= [1, 0, 0, 0, 0], \\ \vec{x}_{water} &= [0, 0, 0, 1, 0]. \end{aligned} \tag{3.5}$$

These five vectors can then be used to generate new distributional vectors, $\vec{y}$, by representing each word by the sum of the vectors $\vec{x}$ of the words with which it co-occurs. If *cat* and *dog* frequently co-occur with each other, and with *pet*, whereas *coastal* mainly co-occurs with *water*, the resulting representations may be similar to

$$\begin{aligned} \vec{y}_{cat} &= [25, 0, 20, 0, 10], \\ \vec{y}_{dog} &= [30, 0, 10, 0, 20], \\ \vec{y}_{coastal} &= [0, 10, 0, 100, 0]. \end{aligned} \tag{3.6}$$

In this representation, *cat* and *dog* are more similar to one another than they are to *coastal*, which is exactly what we want. A visualisation of such a word space is given in Figure 3.2.

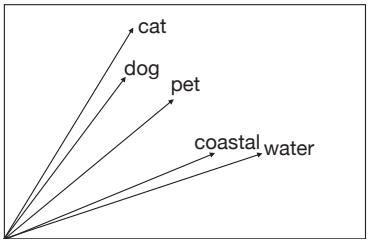

Figure 3.2: An example of a word space.

The type of word representation discussed up until now is also known as a *count*-based representation, as opposed to a *prediction*-based representation (Baroni et al., 2014).[9] Whereas a count-based representation can be seen as counting the words in a given context, a prediction-based representation can be made by attempting to predict that context. Doing this with a neural network, the error obtained when attempting to make such predictions is used to update the representations, until a low error is obtained (see Chapter 2 for details on neural networks). With the entry of the deep learning tsunami on the NLP scene (Manning, 2015), this type of dense word representations has become increasingly popular. The availability of tools implementing such algorithms, such as *word2vec*, undoubtedly helped push the popularity of this approach further. This trend

[9] Whereas Baroni et al. (2014) suggest that prediction-based methods outperform count-based ones, Levy and Goldberg (2014b) show that the underlying differences between the approaches are small.

was introduced by Collobert and Weston (2008), Turian et al. (2010), and Collobert et al. (2011), and was further spearheaded by papers such as Mikolov et al. (2013c), which showed that a simple neural model would encapsulate linguistic regularities in its embedded vector space. The now infamous example of this property is shown in a figure in Mikolov et al. (2013c), replicated in Figure 3.3, where the following relation holds

$$\overrightarrow{king} + \overrightarrow{woman} - \overrightarrow{man} = \overrightarrow{queen}. \tag{3.7}$$

In other words, the distance between *man* and *woman* is similar to that between *king* and *queen*, so adding this difference to the vector of *king* results in a vector close to *queen*.

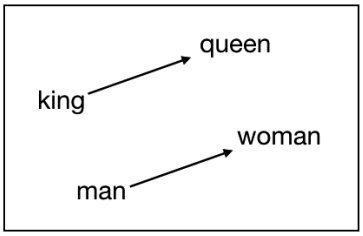

Figure 3.3: A word space in which adding the difference between *woman* and *man* to *king* results in *queen*.

In a prediction-based approach, the terminology used is that a word is *embedded* into n-dimensional space. Typically, this dimensionality is much lower (e.g. around 100) than what is generally used in count-based approaches (e.g. around 1000). In the case of neural networks, these embeddings can be trained together with the rest of the network. This results in a matrix of word vectors in which words with similar properties (under the task at hand) are close to one another.

Distributional vs. Distributed representations

An useful distinction to make is that of *distributed* vs. *distributional* representations. A distributional word representation is based on a co-occurrence matrix, taking advantage of the distributional hypothesis. Similarities between the resulting distributional word representations thus represent the extent to which they co-occur, and therefore also their semantic similarity. A distributed representation, on the other hand, is simply one that is continuous. That is to say, a word is represented by a dense, real-valued, and usually low-dimensional vector. Such representations are generally known as word embeddings, with each dimension representing some latent feature of the word at hand. One way to remember this is that such representations are *distributed* across some n-dimensional space. The first representations we saw were therefore distributional, but not distributed (Turian et al., 2010). The word embeddings, on the other hand, can be said to be both.

Bilingual and Multilingual Word Representations

Going from monolingual to bilingual word representations has been the subject of much attention in recent years. One of the first approaches to bilingual word representations was shown by Klementiev et al. (2012), followed by work such as Wolf et al. (2014), and Coulmance et al. (2015). Parallel to approaches which aim at making good multilingual embeddings, are attempts at producing better monolingual embeddings by exploiting bilingual contexts, as in Guo et al. (2014), Šuster et al. (2016), and Šuster (2016).

In essence, the approaches to building such representations can be divided up into several categories. Cross-lingual mapping can be done by first learning monolingual embeddings for separate languages, and then using a bilingual lexicon to map representations from one space to the other (Mikolov et al., 2013b). Another ap-

proach is to mix contexts from different languages, and training pre-existing systems, such as word2vec, on this mixed data (Gouws and Søgaard, 2015). The approach under consideration in this thesis is based on exploiting parallel texts, by jointly optimising a loss function when predicting multilingual contexts (Guo et al., 2016).

The true power of multilinguality is not unlocked until we can consider an arbitrary number of languages at a time. Whereas bilingual word representations only encode two languages, a multilingual word space contains representations from several languages in the same space. As before, we here also have the property that words with similar meanings are close to one another irrespective of the language (see Figure 3.4).

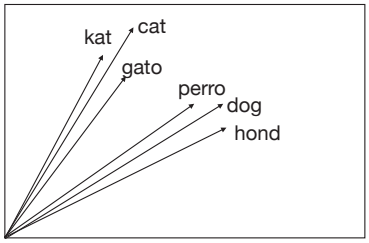

Figure 3.4: An example of a multilingual word space.

One such is the multilingual skip-gram model, as outlined by Guo et al. (2016).[10] As a variant of this model is used in Part III and Part IV, we will now cover this in more detail.

[10]The skip-gram method to create word embeddings, in which a neural network attempts to predict the context of a word, is not to be confused with skip-grams in the sense of n-grams which are not necessarily consecutive. In the second sense, we can define a k-skip-n-gram as a sequence of length n, in which words occur at distance k from each other. In this thesis, only the first sense is of importance.

Multilingual Skip-gram

The skip-gram model has become one of the most popular manners of learning word representations in NLP (Mikolov et al., 2013a). This is in part owed to its speed and simplicity, as well as the performance gains observed when incorporating the resulting word embeddings into almost any NLP system. The model takes a word w as its input, and predicts the surrounding context c. Formally, the probability distribution of c given w is defined as

$$p(c|w;\theta) = \frac{\exp(\vec{c}^T\vec{w})}{\Sigma_{c\in V}\exp(\vec{c}^T\vec{w})}, \tag{3.8}$$

where V is the vocabulary, and θ the parameters of word embeddings ($\vec{w}$) and context embeddings ($\vec{c}$). The parameters of this model can then be learned by maximising the log-likelihood over (w, c) pairs in the corpus C,

$$J(\theta) = \sum_{(w,c)\in D} \log p(c|w;\theta). \tag{3.9}$$

Guo et al. (2016) provide a multilingual extension for the skip-gram model, by requiring the model to not only learn to predict English contexts, but also multilingual ones. This can be seen as a simple adaptation of Firth (1957, p.11), i.e., you shall know a word by the *multilingual* company it keeps. Hence, the vectors for, e.g., *dog* and *perro* ought to be close to each other in such a model. This assumes access to multilingual parallel data, as word alignments are used in order to determine which words comprise the multilingual context of a word.

Formally, the learning objective in multilingual skip-gram is de-

fined in Guo et al. (2016) as

$$
\begin{aligned}
J &= \alpha \sum_{l \in L} J_{mono_l} + \beta \sum_{l \in L, \{EN\}} J_{bi_l, EN} \\
J_{mono_l} &= \sum_{(w,c) \in D_{l \leftrightarrow l}} \log p(c|w;\theta) \\
J_{bi_l, EN} &= \sum_{(w,c) \in D_{l \leftrightarrow EN}} \log p(c|w;\theta),
\end{aligned}
\tag{3.10}
$$

where L denotes the set of all languages, and α and β are weight parameters for the monolingual and bilingual contexts, respectively. In our work, however, we do not rely on always using English as a pivot, and rather use all bilingual pairings to generate contexts. In other words, we also predict the French context based on the Spanish word, and vice versa, rather than only predicting from or to English. This is visualised in Figure 3.5, in which the dashed lines indicate the additional predictions made using the loss described here, and used in Bjerva and Östling (2017a).

Formally, the joint objective function used here is defined as

$$
\begin{aligned}
J &= \alpha \sum_{l \in L} J_{mono_l} + \beta \sum_{l_1 \in L} \sum_{l_2 \neq l_1 \in L} J_{bi_{l_1}, bi_{l_2}} \\
J_{mono_l} &= \sum_{(w,c) \in D_{l \leftrightarrow l}} \log p(c|w;\theta) \\
J_{bi_{l_1}, bi_{l_2}} &= \sum_{(w,c) \in D_{l_1 \leftrightarrow l_2}} \log p(c|w;\theta).
\end{aligned}
\tag{3.11}
$$

3.3 Outlook

In the first part of this chapter, we considered multitask learning, which is the focus of Part II of this thesis. We will first see a case study, in which a MTL paradigm is shown to improve performance on two sequence labelling tasks. Then we turn to a more theoretical investigation into why this is the case.

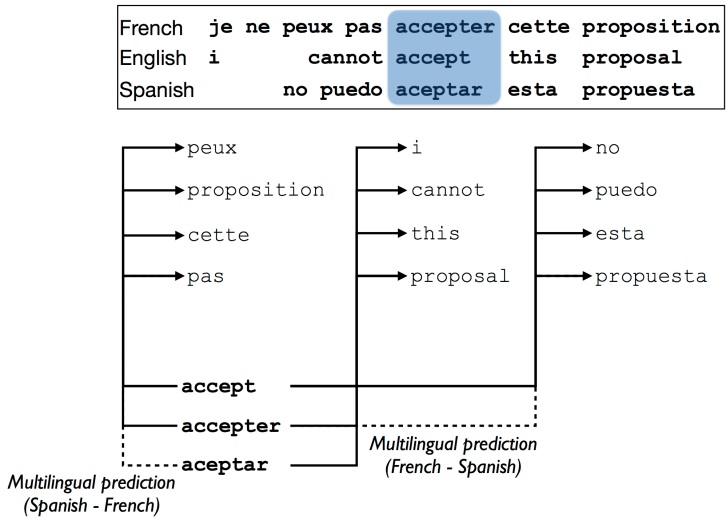

Figure 3.5: Multilingual skip-gram utilising multilingual contexts. Dashed lines indicate the additions of our loss function, i.e., predictions between every language pair.

Following this, Part III also begins with a case study on multilinguality in a single NLP task. The subsequent chapter then includes an empirical study of multilinguality in several tasks, and looks at change in performance when multilinguality is employed.