information

MDPI

Article

Transfer Learning for Named Entity Recognition in Financial and Biomedical Documents

Sumam Francis [1], Jordy Van Landeghem [2] and Marie-Francine Moens [1]

[1] Department of Computer Science, Language Intelligence & Information Retrieval Lab (LIIR), 3000 KU Leuven, Belgium
[2] Contract.fit, 1000 Brussels, Belgium
* Correspondence: sumam92@gmail.com

Received: 30 May 2019; Accepted: 25 July 2019; Published: 26 July 2019

Abstract: Recent deep learning approaches have shown promising results for named entity recognition (NER). A reasonable assumption for training robust deep learning models is that a sufficient amount of high-quality annotated training data is available. However, in many real-world scenarios, labeled training data is scarcely present. In this paper we consider two use cases: generic entity extraction from financial and from biomedical documents. First, we have developed a character based model for NER in financial documents and a word and character based model with attention for NER in biomedical documents. Further, we have analyzed how transfer learning addresses the problem of limited training data in a target domain. We demonstrate through experiments that NER models trained on labeled data from a source domain can be used as base models and then be fine-tuned with few labeled data for recognition of different named entity classes in a target domain. We also witness an interest in language models to improve NER as a way of coping with limited labeled data. The current most successful language model is BERT. Because of its success in state-of-the-art models we integrate representations based on BERT in our biomedical NER model along with word and character information. The results are compared with a state-of-the-art model applied on a benchmarking biomedical corpus.

Keywords: deep learning; entity extraction; named entity recognition; transfer learning; fine-tuning; minimum training data

1. Introduction

Lack of sufficient annotated data often limits the applicability of deep learning (DL) models to real life problems. However, efficient transfer learning (TL) strategies help to utilize valuable knowledge learned with sufficient data in one task (source task) and transfer it to the task of interest (target task).

In this work we focus on a generic named entity recognition (NER) system that uses the representation learning capability of deep neural networks. NER refers to a subtask of information extraction in which entity mentions in an unstructured text are semantically labeled into pre-defined categories (e.g., in the sentence "Benjamin Franklin is known for inventing the lightening rod", "Benjamin Franklin" has to be labeled with the tag *Person*). In this paper NER is evaluated with two main use cases—extraction of entity names from financial documents and from biomedical documents.

The aim of the first use case is to develop a generic NER deep learning system that is capable of recognizing entities in business documents including invoices, business forms and emails. Some examples of important named entities used in the financial documents are *invoice sender name*, *invoice number*, *invoice date*, *International Bank Account Number* (*IBAN*), *amount payable* and *structured communication*. The overall goal of this use case is to research and test prototypes for a self-learning SaaS platform for simplification of data-intensive customer interactions in industries such as energy, telecommunications, banking or insurance. The format of business documents from different service providers varies widely with respect to the information contained within it, the locations where it occurs, the formatting used, terminology used, and variations in context.

The growing biomedical literature requires efficient methods of text mining and information extraction tools that can extract structured information from the text for further analysis. The second use case has as objective the development and application of a generic NER system that recognizes biomedical entities like *genes, proteins, chemicals, diseases* and the like from biomedical text sequences.

In both cases we want to port a model learned in one domain to a target domain that is characterized by a limited number of labeled training examples. Multiple additional challenges apply. The difficulty posed by the documents in the financial domain is the noisy non-segmented OCRed text which has to be used as the training data. The documents in the biomedical domain are characterized by spelling variants of the mentioned entities.

Neural network based deep learning models have become popular for processing unstructured (e.g., language) and semi-structured data. Deep learning is concerned with automatically learning the representations of the inputs where patterns at different levels of detail can be recognized and represented, a process which is referred to as representation learning. In particular, recurrent neural networks (RNNs) are capable of learning contextualized representations of text sequences. Information extraction from text using deep learning avoids the need for humans to formally specify knowledge, and even avoids the need for feature engineering. Most state-of-the-art named entity recognition (NER) systems use word level and character level representations (embeddings) to perform named entity extraction and classification (dos Santos and Guimarães [1]).

A word or character is mapped to a continuous vector representation (embedding) that captures the context of the word and character respectively. While word based models need accurate token-level segmentation, character level models have the ability to perform accurate labelling of tokens or character units without the need for a-priori word segmentation (Kann and Schütze [2]).

The aim is that the extraction system can be trained with few manually annotated training data, especially when a trained model needs to be ported to a novel domain (e.g., another provider of documents to be analyzed, other entity classes to be recognized).

We witness a rising interest in language models to improve NER in a transfer learning setting. The current most successful language model is BERT (Devlin et al. [3]). We integrate BERT representations to analyse its impact when using it along with separate character and word representations.

The research questions posed in the paper are:

- How to train models (e.g., character based neural networks) with small sized semi-structured labeled datasets (e.g., contracts and invoices)?
- What is the impact of transfer learning when a NER model that is trained on a source dataset labeled with one type of entities is ported to a target dataset to be labeled with another but related entity type and only few labeled documents in the target domain are available?
- What is the impact of using the BERT language model along with character and word character level representations?

The major contributions presented in the paper are: (a) Development of an efficient pipeline for entity extraction from financial documents that is efficient at runtime and easily parameterizable, and which includes efficient segmentation, filtering of entity elements and finally labelling and extraction; (b) Effective document segmentation by splitting a document text into meaningful and contextual sub-segments; (c) Easy porting from models trained on one task to another task using transfer learning; and (d) Comparison of the impact of integrating the BERT language model with the character and word level representations. The impact of using limited training data in the target domain has been investigated by using different percentages of target training examples.

2. Related Work

NER is a well studied task in natural language processing, yet it has mostly been studied in the frame of recognition of entity types that refer to persons, organizations, locations, or dates. Traditionally NER is seen as a sequence labeling task in which word tokens are labeled with the corresponding named entity class (often using a BIO-format referring to respectively the Beginning, Intermediate or Outside class token). There has been a lot of previous work on optimizing neural architectures for sequence labeling. Collobert et al. [4] introduced one of the first task-independent neural sequence labeling models using a convolutional neural network (CNN). They were able to achieve good results on other sequence labelling tasks such as Part-of-Speech (POS) tagging, chunking, and semantic role labeling (SRL) without relying on hand-engineered features. Huang et al. [5] describe a bidirectional long short-term memory network (LSTM) model, a type of recurrent neural network (RNN), with a conditional random field (CRF) layer, which includes hand-crafted features specialised for the task of named entity recognition. The architecture of our baseline sequence labelling word model for biomedical data uses a similar bidirectional LSTM (BiLSTM) and CRF with batch normalization.

Character level models are capable of capturing morpheme patterns and hence improve the generalisation of the model on both frequent and unseen words. Kim et al. [6] have implemented a language model with a convolutional neural network (CNN) and LSTM architecture using characters as input while the predictions are made at the word level. We use similar neural architectures when extracting named entities from the financial and biomedical documents. For the biomedical domain, we chose to implement word based models and also combinations of word and character based models firstly due to the availability of pretrained word embeddings trained on huge biomedical corpora and general Wikipedia text which considerably mitigates the problem of out-of-vocabulary (OOV) words. Character level models are semantically more void, yet morphologically more informed, and thus word based and character based models are hypothesized to perform better for biomedical domain texts.

Cao and Rei [7] proposed a method for learning both word embedding and morphological segmentation with a bidirectional RNN over characters. Ling et al. [8] have proposed a neural architecture that replaces word embeddings with dynamically constructed character based representations. Lample et al. [9] describe a model where the character level representation is combined with a word embedding through concatenation. Miyamoto and Cho [10] have also described a related method for the task of language modelling, combining character and word embeddings using gating.

Transfer learning is a machine learning technique where a model pre-trained for one task is reused for a second related task that has less labeled training data. Transfer learning in general improves the generalisation of the model. In this paper we apply transfer learning in two different use cases, that is, NER in financial and biomedical documents. We study how a neural model for NER of one entity type which is structurally or contextually similar to some other entity type can avoid the need for training the target model from scratch. For example, we transfer knowledge from a trained model that has learned to properly extract the entity *invoice date* to the extraction of a related entity *expiration date*. This is a case of learning a new task where the character pattern of the new entity type is similar but the context in which this entity occurs is different.

Model based transfer uses the similarity and relatedness between the source task and the target task by modifying the model architectures, training algorithms, or feature representations. For example, Ando and Zhang [11] propose a transfer learning framework that shares structural parameters across multiple tasks, and improve the performance on various tasks including NER. Collobert et al. [4] present a task-independent CNN and employ joint training to transfer knowledge from NER and POS tagging to sentence chunking. Peng and Dredze [12] study transfer learning between NER and word segmentation in Chinese based on RNN architectures. We have exploited model based transfer learning in our models to improve the target task models by investigating different levels of freezing and fine-tuning for the target architecture based on Yang et al. [13], Rodriguez et al. [14] and Lee et al. [15].

We witness an interest in language models to improve NER as a way of coping with limited labeled data. The current most successful language model is BERT (Devlin et al. [3]). BERT makes use of a Transformer architecture, which integrates an attention mechanism that learns contextual relations between words (or sub-words) in a text.

BioBERT (Lee et al. [16]) provides a pre-trained biomedical language model trained with the BERT architecture that can be used in various biomedical text mining tasks such as NER. Because of its success in state-of-the-art models we integrate embeddings obtained with the BERT language model in our NER models for biomedical texts.

3. Methodologies

3.1. Character Level Neural Network

The basic character level neural network (see Figure 1a) receives a sequence of characters $(c_1, \ldots, c_T)$ as input, and predicts a label corresponding to each of the input characters. The characters are either encoded as one-hot encodings or are mapped to randomly initialized character embeddings $(y_1, \ldots, y_T)$. Next, the one-hot encoding or the character embedding is given as input to a bidirectional LSTM whose forward and backward components moving in opposite directions through the text are concatenated and thus create context-specific representations.

$$h_t^{forward} = LSTM(y_t, h_{t-1}^{forward}) \tag{1}$$

$$h_t^{backward} = LSTM(y_t, h_{t+1}^{backward}) \tag{2}$$

$$h_t = [h_t^{forward}; h_t^{backward}] \tag{3}$$

Finally, to produce label predictions, we can use either a softmax layer or a conditional random field layer. The softmax calculates a normalised probability distribution over all the possible labels for each word. We can also use a CRF as the output layer, which conditions each prediction on the previously predicted label (Lafferty et al. [17]). In this architecture, the last hidden layer is used to predict probability scores for each character to predict the possible target labels. Finally, the Viterbi algorithm (Sakharov and Sakharov [18]) is used to find an optimal sequence of weights. We call this model *CharacterModel*. To optimise this model we minimise the categorical cross-entropy.

$$E = -\sum_{t=1}^{T} \log\left(P(y_t = l | h_t)\right) \tag{4}$$

where $P(y_t|h_t)$ is the probability of the label of the t^{th} character y_t being l.

3.2. Word Level Neural Network

A basic word level neural network (see Figure 1b) for sequence labeling receives a sequence of word tokens $(w_1, \ldots, w_T)$ as input, and predicts a label corresponding to each of the input tokens. The tokens are first mapped to a vector resulting in a sequence of word embeddings $(x_1, \ldots, x_T)$. Next, the embeddings are given as input to a bidirectional LSTM whose forward and backward components moving in opposite directions through the text are concatenated and thus create context-specific representations. We call this model *WordModel*.

$$h_t^{forward} = LSTM(x_t, h_{t-1}^{forward}) \tag{5}$$

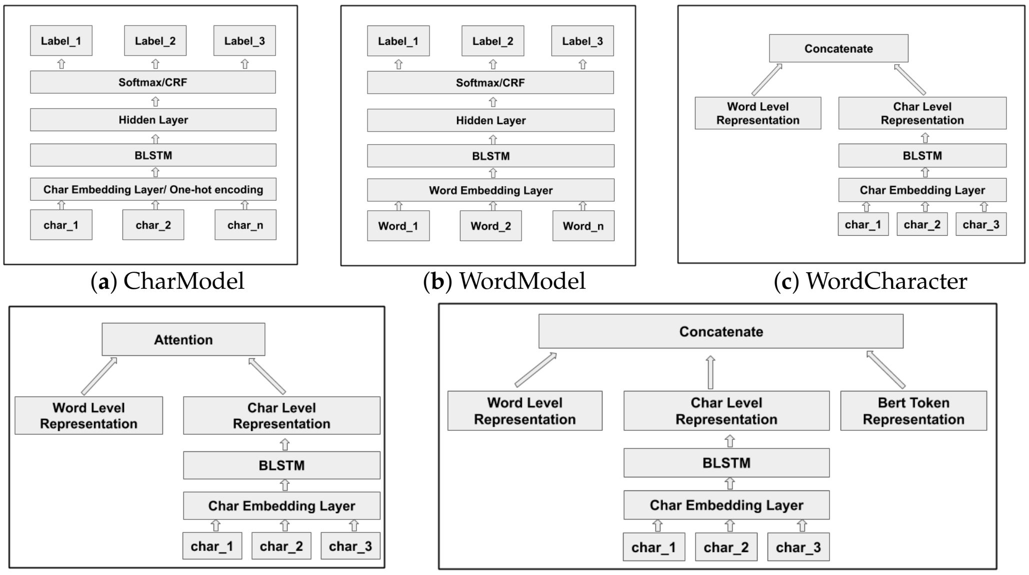
$$h_t^{backward} = LSTM(x_t, h_{t+1}^{backward}) \tag{6}$$

$$h_t = [h_t^{forward}; h_t^{backward}] \tag{7}$$

We add an extra narrow hidden layer on top of the biLSTM, which allows the model to detect higher-level feature combinations.

$$d_t = tanh(W_d h_t) \tag{8}$$

where W_d is the corresponding weight matrix.



(**a**) CharModel (**b**) WordModel (**c**) WordCharacter

(**d**) WordCharacterAttention (**e**) WordCharacterBERT

Figure 1. Architectures of the word, character and BERT level representation models.

Finally, to produce label predictions, we can use either a softmax layer or a CRF layer. Further the Viterbi algorithm (Sakharov and Sakharov [18]) is used to find an optimal sequence of weights. To optimise this model and the models described in the following two sections, we minimise the categorical cross-entropy, i.e., the negative log-probability of the correct labels.

$$E = -\sum_{t=1}^{T} \log\left(P(y_t = l|d_t)\right) \tag{9}$$

where $P(y_t|d_t)$ is the probability of the label of the t^{th} word (y_t) being l.

3.3. Word and Character Level Neural Network

Sequence labeling architectures use word embeddings for capturing similarity but suffer when handling previously unseen or rare words. By adding character level information to such models, the model is able to infer representations for previously unseen words and obtain syntactic information at the morpheme level. Consequently, they can identify morphological similarities between different words. In general, each word is broken down into individual characters, these are then mapped to a sequence of character embeddings $(c_1, \ldots, c_R)$, which are passed through a bidirectional LSTM.

$$h_i^{l,forward} = LSTM(c_i, h_{i-1}^{l,forward}) \tag{10}$$

$$h_i^{l,backward} = LSTM(c_i, h_{i+1}^{l,backward}) \tag{11}$$

We then use the last hidden vectors from each of the LSTM components, concatenate them together and pass the result through a separate non-linear layer. We call this model *WordCharacter model* (see Figure 1c).

$$h_t = [h_i^{l,forward}; h_i^{l,backward}] \tag{12}$$

The output is a joint word representation built from individual characters. We now have two alternative feature representations for each word, where $x(t)$ is an embedding learned on the word level, and $k(t)$ is a representation built from individual characters at the t^{th} word of the input text.

Then according to Lample et al. [9], one possible approach is to concatenate the two vectors and use this as the new word level representation for the sequence labeling model:

$$x_{concat} = [x; k] \tag{13}$$

3.4. Word and Character Level Neural Network with Attention

This architecture combines a character level representation with a word embedding using a gating mechanism (Rei et al. [19]), also referred to as attention. We call this model *WordCharacterAttention* (see Figure 1d).

This allows the model to dynamically decide which source of information to use for each word. Here instead of just concatenating the word and character level representation the two vectors are added together using a weighted sum, where the weights are predicted by a two-layer network.

$$z = \sigma\left(W_z^{(3)}\, tanh\left(W_z^{(1)}x + W_z^{(2)}k\right)\right) \tag{14}$$

where $W^{(1)}$, $W^{(2)}$ and $W^{(3)}$ are weight matrices for calculating z and σ is a sigmoid logistic function. z is the weight matrix between word representation x and character representation k.

$$x_{att} = z \cdot x + (1 - z) \cdot k \tag{15}$$

3.5. Word- and Character Level Neural Network Combined with the BERT Language Model

BERT (Devlin et al. [3]) can be used to obtain pre-trained word representations based on a language model. The BERT model that we use is BERT-Base uncased (https://github.com/google-research/bert) which uses 12 layers of transformer encoders, 768 hidden layers and 12 attention heads. The output of each layer given a word as input can be considered as a word embedding. One of the best performing choices—also used in this paper—is to sum the word embedding of the last 4 layers to obtain the BERT model embedding. To obtain the word embedding from BERT's pre-trained model, we make use of a public resource (https://github.com/imgarylai/bert-embedding).

In our model the neural network receives a sequence of word tokens $(w_1, \ldots, w_T)$ as input, and predicts a label corresponding to each of the input tokens. The tokens are first mapped to a vector resulting in a sequence of word embeddings $(x_1, \ldots, x_T)$. The tokens are also mapped to a sequence of vectors using BERT embeddings. The character level representations of the words are also used. The word representation, character representation and BERT representation are concatenated. Next, the concatenated embedding is given as input to a bidirectional LSTM whose forward and backward components moving in opposite directions through the text are concatenated and thus create context-specific representations. We call this model *WordCharacterBERT* (see Figure 1e). $x(t)$ is an embedding learned on the word level, $k(t)$ is a representation built from individual characters at the t^{th} word of the input text and $z(t)$ is an embedding learned by the BERT model.

Then by concatenating the three vectors we obtain the new word level representation that will be used in the sequence labeling NER model (see below):

$$x_{concat} = [x; k; z] \tag{16}$$

3.6. Transfer Learning

Most deep learning models are specialized in a particular domain or even a specific task for which they are trained. Transfer learning goes beyond specific tasks and domains. It leverages knowledge from pre-trained models and uses the acquired knowledge to solve new problems. In task related transfer learning, we use the knowledge in the form of features or weights learned from previously trained models for training newer models for the target task which is related. If we have significantly more labeled data for task $T1$, we utilize its model learned from this data and generalize this knowledge (features, weights) for task $T2$ (which has significantly less labeled data). In general instead of training a deep learning model on the target task from scratch, transfer learning allows us to utilize a network trained on a different task and use it on a target task by adapting it. There are different transfer learning strategies that are used in the academic community.

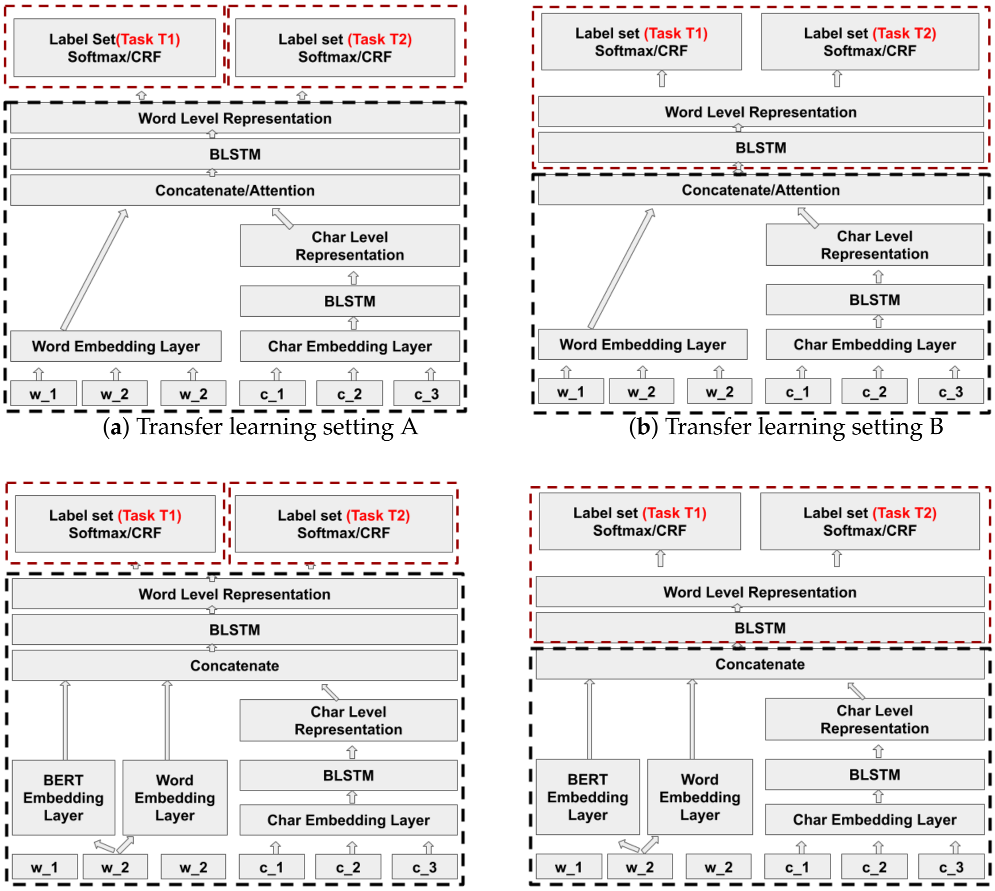
Let $D_s = (X_j^s, Y_j^s)$ where $j = 1$ to N be the training set of N samples from the source dataset and $D_t = (X_j^t, Y_j^t)$ where $j = 1$ to M be the training set of M samples from the target domain. The Bi-LSTM encodes word and character embeddings into hidden vectors $H = (h_1, h_2, \ldots, h_t)$. The final CRF layer decodes hidden vectors H to a label sequence $y = (y_1, y_2, \ldots, y_n)$. Our goal is to improve label prediction accuracy on the target domain D_t by using the knowledge learned from the source domain D_s.

Using pre-trained models as *Feature Extractors* is one common strategy of using transfer learning. Deep learning systems and models are layered architectures that learn different features at different layers. This layered architecture allows us to utilize a pre-trained network without its final layer as a fixed feature extractor for other tasks. Recently language models pre-trained on domain data have been used as feature extractors for different NLP tasks (Devlin et al. [3], Lee et al. [16], Howard and Ruder [20]). In this paper we use a pretrained BERT model for obtaining BERT token embedding (see Figure 2c). The Bi-LSTM further encodes the concatenation of the BERT, word and character representations into hidden vectors. The final CRF layer decodes the hidden vectors to a label sequence.

Using a pre-trained model for *Fine-Tuning* is another setting for transfer learning. Erhan et al. [21] and Collobert et al. [4] have shown that initializing parameters with values of other supervised or unsupervised models often improves model convergence. In this framework of transfer we first train a model for a source task and use the learned parameters to initialize the model parameters for training the target task. This strategy is applicable for both same label set as well as disparate label set transfer. We transfer entire network parameters if there exists a label mapping between source and target label sets, otherwise we only share model parameters up to the second last layer of the network. The level of freezing and fine-tuning of different layers in the base model depends on the target task. Another common setting is using both source and target domain data in a multitask setting.

For sequence labelling tasks, the two tasks can have label sets that can be mapped (similar) to each other or very different label sets. If the two tasks have similar label sets, then we share all the model parameters and feature representations in the neural networks, including the word and character embeddings, the word level layer, the character level layer, and the CRF or softmax layer similar to Giorgi and Bader [22] and Yang et al. [13].

- *Transfer learning setting A (TL-A)*: If the two tasks make use of different label sets, then we remove the parameter sharing in the CRF or softmax layer. So now each task learns a separate CRF or softmax layer (see Figure 2a). The impact of freezing and fine-tuning different layers in the base model can also be studied.
- *Transfer learning setting B (TL-B)*: In Figure 2b, only the character and word embedding are shared, all other parameters are fine-tuned. This can be used to study the impact of using certain types of embeddings.
- *Transfer learning setting C (TL-C)*: In Figure 2c we study the impact of using the BERT embeddings along with character and word-character level representations when transferring from one task to another. Here each task learns a separate CRF or softmax layer. All other layers are shared.
- *Transfer learning setting D (TL-D)*: In Figure 2d we again use the BERT embedding along with the character and word-character level representation, and only the resulting concatenated representation at the word level is shared.


Label Set(Task T1) Softmax/CRF
Label set (Task T2) Softmax/CRF
Word Level Representation
BLSTM
Concatenate/Attention
Char Level Representation
BLSTM
Word Embedding Layer
Char Embedding Layer
w_1
w_2
w_2
c_1
c_2
c_3
Label set (Task T1) Softmax/CRF
Label set (Task T2) Softmax/CRF
Word Level Representation
BLSTM
Concatenate/Attention
Char Level Representation
BLSTM
Word Embedding Layer
Char Embedding Layer
w_1
w_2
w_2
c_1
c_2
c_3
Label set (Task T1) Softmax/CRF
Label set (Task T2) Softmax/CRF
Word Level Representation
BLSTM
Concatenate
Char Level Representation
BLSTM
BERT Embedding Layer
Word Embedding Layer
Char Embedding Layer
w_1
w_2
w_2
c_1
c_2
c_3
Label set (Task T1) Softmax/CRF
Label set (Task T2) Softmax/CRF
Word Level Representation
BLSTM
Concatenate
Char Level Representation
BLSTM
BERT Embedding Layer
Word Embedding Layer
Char Embedding Layer
w_1
w_2
w_2
c_1
c_2
c_3

(**a**) Transfer learning setting A

(**b**) Transfer learning setting B

(**c**) Transfer learning setting C

(**d**) Transfer learning setting D

Figure 2. Transfer learning settings—The black stripped box indicates the shared parameters and the red stripped box indicates the fine-tuned parameters for a specific task.

For fine-tuning the base model, some implementation tricks mentioned in Howard and Ruder [20] are used. Discriminative fine-tuning ensures that instead of using the same learning rate for all layers of the model, each layer is fine-tuned with different learning rates. This is done by first choosing the learning rate ηl of the last layer, by fine-tuning only this last layer on the development set and then using $\eta l - 1 = \eta l/2.6$ as the learning rate for lower layers. This choice of learning rate was inspired by the paper Howard and Ruder [20] where they choose this rate heuristically after performing several experiments for fine-tuning the language model. We used a learning rate of ηl = 00000.1 for the last layer. For subsequent layers the learning rate was chosen according to the learning rate scheme defined above. Gradual unfreezing is another trick which ensures that the pre-trained model is gradually unfreezed starting from the last layer. Rather than fine-tuning all layers at once, which can result in huge forgetting, we gradually unfreeze the model starting from the last layer as this contains the least general knowledge. Hence we first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We further unfreeze the next lower frozen layer and repeat this, until all layers are fine-tuned.

4. Experiments and Discussion

4.1. Use Case 1: Financial NER Extraction

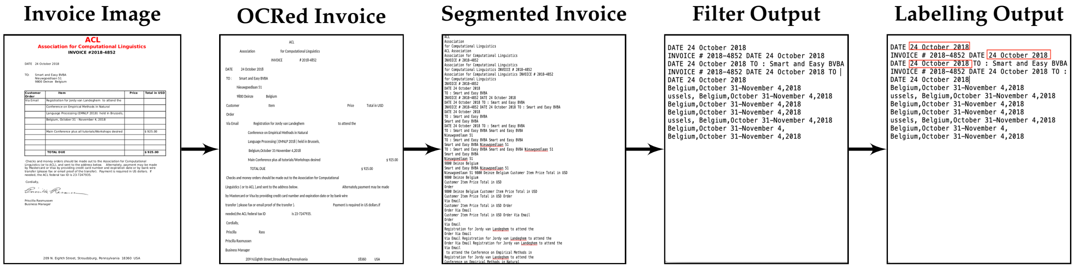
4.1.1. Dataset

The financial dataset is comprised of 3000 documents which are mainly invoices from different service providers like insurance, telecommunications, banking and tax companies. Typical entities of interest to be extracted are the *IBAN* of the beneficiary, *invoice number*, *invoice date* and *due date*, various amounts like the *total inclusive amount*, *total exclusive amount* and *total VAT amount*, *structured communication*, *company name*, *company address* and other related fields required to book or pay an invoice (see Table. 1). The patterns that contribute to the recognition of the entities include the character patterns of the entity itself and the patterns of the context of the entity.

4.1.2. Data Preprocessing

We first apply optical character recognition (OCR) to convert the documents into a textual representation with the additional goal of preserving the original document layout as well as possible (see Figure 3).

The character vocabulary is constrained to alpha-numeric characters and some special symbols (such as valuta symbols).



```
DATE 24 October 2018
INVOICE # 2018-4852 DATE 24 October 2018
DATE 24 October 2018 TO : Smart and Easy BVBA
INVOICE # 2018-4852 DATE 24 October 2018 TO |
DATE 24 October 2018
Belgium,October 31-November 4,2018
ussels, Belgium,October 31-November 4,2018
Belgium,October 31-November 4,2018
Belgium,October 31-November 4,2018
ussels, Belgium,October 31-November 4,2018
Belgium,October 31-November 4,
Belgium,October 31-November 4,2018
```

```
DATE 24 October 2018
INVOICE # 2018-4852 DATE 24 October 2018
DATE 24 October 2018 TO : Smart and Easy BVBA
INVOICE # 2018-4852 DATE 24 October 2018 TO :
DATE 24 October 2018|
Belgium,October 31-November 4,2018
ussels, Belgium,October 31-November 4,2018
Belgium,October 31-November 4,2018
Belgium,October 31-November 4,2018
ussels, Belgium,October 31-November 4,2018
Belgium,October 31-November 4,
Belgium,October 31-November 4,2018
```

Figure 3. Pipeline for NER in financial documents.

Table 1. An example of common entity formats that appear in the invoices.

Entity	Example
IBAN	BE01 3456 7890 1234
Invoicedate	03/04/19
Totalinclusive	€56.78
Invoicenumber	F12345678
Structured-communication	+++170/0768/36648+++

4.1.3. Line Segmentation

The document-text line output is further processed by the document line segmentation algorithm which segments the document into meaningful line segments (see Figure 3). We have devised a parameterizable (e.g., minimum and maximum segment length) *segmentation algorithm* that exploits the document's layout structure.

4.1.4. Filter Network

The filter network is a binary classifier which filters out the segments having high probability of containing the element of interest from the segmented invoices. Its neural network architecture comprises of an input layer, 2 stacked Bidirectional LSTM layers and a fully connected layer with batch normalization applied between the layers (see Figure 4). The line sequences are read at the character level. The input is a one hot encoding or randomly initialized character embedding. The advantages of using character level deep learning models include alleviating vocabulary problems, removing the computational bottleneck at the output of our model, and being resilient to spelling/OCR mistakes and to rare or unseen codes or words. The morphological information captured using character models are particularly beneficial in invoices where we mainly deal with entities with near-unique patterns.

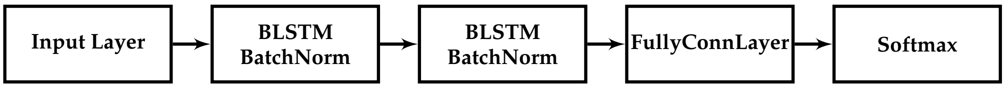


Figure 4. Architecture of the filter network.

At the output layer a softmax activation outputs the probability of the correct class. All segmented text lines which have prediction probability greater than 0.5 are filtered out and passed to the next neural network.

4.1.5. NER: Sequence Labelling

The second neural network aims at predicting the named entity class label of each character. We have opted for the BIO-labelling scheme for encoding the target labels (see Table 2).

Table 2. An example of a BIO-tagged named entity.

Textline with Entity	Example BIO-Labelling
IBAN No: BE01345678901234	OOOOOOOOOOBIIIIIIIIIIIIII

The architecture of the sequence labelling network comprises of an input layer, 2 stacked bidirectional LSTM layers with batch normalization applied in between (see Figure 5) and finally a CRF (conditional random field) layer, which simulates label dependencies in the output.

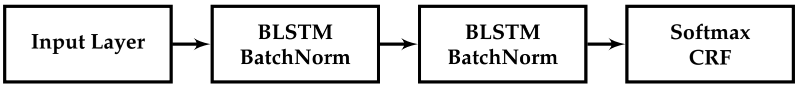


Figure 5. Architecture of the NER sequence labeling.

Finally, the element of interest and its span are extracted from the invoice and tagged with the predicted entity class on the basis of the start span and end span of the predicted BIO tags. We use beam search decoding to find the most probable tag sequence of an entity.

4.1.6. Evaluation of the Extracted Entities

The extracted entities are compared with standardized database entries of these entities, which follow a predefined scheme. Table 3 shows the difference between the extracted entities and their gold database entries.

Table 3. Examples that illustrate the difference between a gold database entry and its occurrence(s) in a document's text.

Entity Label	Gold Database Entry	Document Occurrence
IBAN	BE04 7123 7854 0002	be047123 7854 00 02
Communication	+++518/0530/28569+++	+ + + 5 1 8105 3 0 2 8 5 6 9
Invoice date	20/10/2018	20 oct '18 | 20 October 2018

Provided the variance in the structure and appearance of an entity in the financial documents, string matching techniques would not suffice to properly evaluate the extractions. Hence, in our experiments the predicted and gold financial entities were manually evaluated.

4.1.7. Experimental Setup

The first set of experiments for the financial use case was designed to check the end-to-end performance of models for different entities in an invoice. For purpose of comparison we report results of four types of entities. *IBAN* generally follows a handful of pattern variations that are often slightly modified by OCR noise. Extraction of this element is less complex given the typical character patterns of IBAN codes. *Invoice date* has predictable patterns, yet to distinguish this entity from other dates requires context analysis. *Total inclusive amount* is foremost dependent on context to disambiguate with other types of amounts but has less variance in character patterns. *Invoice number* is certainly the most difficult entity to recognize due to its random alphanumeric patterns and is thereby easily confused with other entities occurring in an invoice such as *order number*.

As mentioned above NER in the financial use case relies on character based models. The character embedding is set to length 25 and initialized randomly. The LSTM layer size is set to 128 in each direction for both the LSTM layers. For the filter network, the fully connected feedforward layer has a size of 60. Parameters are optimized using Adam with learning rate of 0.0001 and sentences are grouped into batches of size 128. Performance on the development set is measured at every epoch and training is stopped if performance had not improved for 12 epochs. The best-performing model on the development set is then used for evaluating the test set.

All datasets mentioned in the financial domain comprise of 3000 documents of which 60% are used as training set, 25% as test set and 15% as development set. For transfer learning we have used a dataset tagged with the entity class *invoice date* and one with the entity class *expiration date*, each comprising of 3000 documents. In the transfer learning task the number of training examples (in training and development set) for the target NER task may be reduced (see below).

4.1.8. Results

In Table 4 we report the test results of the NER as well as the results of the filter network, which forms an important step in the pipeline.

Table 4. Results of NER in the financial use case in terms of F1.

Experiment	Test Filter F1	Test NER F1
IBAN	0.99	0.98
Invoice date	0.92	0.85
Total inclusive	0.94	0.82
Invoice number	0.96	0.88

The transfer learning regards the sequence labeling of the NER task. In the transfer learning experiments we evaluate the influence of using extraction patterns of related entities in an invoice. Both *invoice date* and *expiration date* are related entities with respect to their character patterns while the context in which they occur is quite different and crucial to differentiate their type.

Without transfer: In this experiment we use the target domain labeled data only. This is used as a baseline. In Table 5, we illustrate the base model results of both *invoice date* and *expiration date*.

Table 5. Results in terms of F1 scores of the segment filtering and NER sequence labeling (**Without transfer**).

Experiment (Label-Type)	Test Filter F1	Test NER F1
Invoice date	0.92	0.85
Expiration date	0.87	0.82

Fine-tuning the pre-trained model: The primary purpose of this experiment is to assess to what extent transfer learning improves the performances on the target NER task. We experiment with different target train set sizes to understand how many labeled examples of the target NER are needed in this transfer learning task (Table 6).

Analyze transferability of different layers: In this experiment, we analyze the importance of each parameter of the pre-trained DNN model in the transfer learning. Instead of transferring all the parameters, we experiment with transferring different combinations of parameters. The goal is to understand which components of the pre-trained model are the most important to transfer. The lowest layers usually represent task-independent features whereas the topmost layers are more task-specific.

In order to understand the impact of using transfer learning, we use a pre-trained model trained for *invoice date* recognition and fine-tune it when extracting the *expiration date* in the target domain. The different levels of freezing used are:

1. *Lastlayer-Finetuning*: Fine-tuning only the last layer weights, i.e., only the CRF or softmax layer is fine-tuned while all the previous layers weights are kept fixed or frozen.
2. *SecondBiLSTM-Finetuning*: Fine-tuning all weights until the second BiLSTM layer representations including the CRF layer. The previous layers weights are kept fixed or frozen.

Table 6. ***Invoice date* to *expiration date***: F1 score of recognition of *expiration date* in the test set by transfer learning from a base model trained with *invoice date* labels. Results at different levels of freezing trained with different percentages of labeled target data are presented.

Setting	**Model**	**Fine-Tuning**	**100%**	**75%**	**50%**	**25%**	**10%**
Without transfer	Filter		0.87	0.86	0.85	0.83	0.81
With transfer (TL-A)	Filter	Lastlayer-Finetuning	0.88	0.87	0.86	0.85	0.85
With transfer (TL-B)	Filter	SecondBiLSTM-Finetuning	0.88	0.87	0.86	0.85	0.85
Without transfer	NER		0.82	0.81	0.80	0.79	0.77
With transfer (TL-A)	NER	Lastlayer-Finetuning	0.86	0.85	0.84	0.83	0.80
With transfer (TL-B)	NER	SecondBiLSTM-Finetuning	0.87	0.85	0.84	0.83	0.80

Similarly in Table 7 we study the impact of using transfer learning from a pre-trained *expiration date* model and fine-tune it using *invoice date* labels. This is to see if both entity types are able to transfer knowledge from and to.

Table 7. ***Expiration date* to *invoice date***: F1 score of recognition of *invoice date* in the test set by transfer learning from base model trained with *expiration date* labels. Results at different levels of freezing trained with different percentages of labeled target data are presented.

Setting	**Model**	**Fine-Tuning**	**100%**	**75%**	**50%**	**25%**	**10%**
Without transfer	Filter		0.92	0.91	0.89	0.87	0.84
With transfer (TL-A)	Filter	Lastlayer-Finetuning	0.92	0.91	0.90	0.89	0.88
With transfer (TL-B)	Filter	SecondBiLSTM-Finetuning	0.93	0.91	0.90	0.89	0.88
Without transfer	NER		0.85	0.85	0.82	0.81	0.79
With transfer (TL-A)	NER	Lastlayer-Finetuning	0.88	0.88	0.87	0.86	0.85
With transfer (TL-B)	NER	SecondBiLSTM-Finetuning	0.89	0.88	0.87	0.86	0.86

As can be seen from Tables 6 and 7, transfer learning improves the F1-scores compared to training only with the labeled target dataset without transfer. Even when the number of training samples used for the target dataset decreases, the F1-scores remain considerably high. This implies that the representations learned from the source dataset and task are efficiently transferred and exploited for performing well in the target domain.

4.2. Use Case 2: Biomedical NER

4.2.1. Dataset

In the biomedical use case we rely on two main benchmarking datasets—BC2GM (Smith et al. [23]) and BioNLP09 (Kim et al. [24]). The BioCreative II Gene Mention (BC2GM) corpus consists of 20,000 sentences from biomedical publication abstracts and is annotated for mentions of the names of genes and related entities using a single named entity class *GENE*. The different named entity tags used in the dataset are *B-GENE*, *I-GENE* and *O* tag. The BioNLP 2009 GENIA shared task set corpora comprises of 10,000 sentences and is annotated for mentions of the names of proteins and related entities that appear in the biomedical literature using a single named entity class *PROTEIN*. The different named entity tags used in the dataset are *B-PROTEIN*, *I-PROTEIN* and *O* tag.

These datasets are already tokenized in words. Both datasets are publicly available and can be downloaded from a public resource (https://github.com/cambridgeltl/MTL-Bioinformatics2016).

4.2.2. Data Preprocessing

All digits are replaced with the character '0'. Any word that occurs only once in the training data is replaced by the generic OOV (out of vocabulary) token when computing the word embedding, but is still used in the character level components. The word embeddings are initialised with publicly available pre-trained vectors, created using word2vec (Mikolov et al. [25]), and then fine-tuned during model training. For the biomedical datasets we use 200-dimensional vectors trained on PubMed. The embedding for characters are set to length 50 and initialized randomly.

4.2.3. Experimental Setup

For NER in biomedical documents, we have used four models as base models, which are described in detail in Section 3.

- *WordModel* where we build word level representations only based on traditional word2vec embeddings.
- *WordCharacter* where we build word level representations by concatenating traditional word2vec embeddings and character embeddings.
- *WordCharacterAttention* where we build word level representations by concatenating traditional word2vec embeddings and character embeddings in addition to using attention which acts as a gating mechanism.
- *WordCharacterBERT* where we build word level representations by concatenating traditional word2vec embeddings, character embeddings and BERT pretrained embeddings.

The LSTM layer size was set to 200 in each direction for both word and character level layers. The hidden layer has size 50. CRF was used as the output layer for all the experiments. Parameters are optimized using Adam with default learning rate of 0.0001 and sentences are grouped into batches of size 64. Performance on the development set is measured at every epoch and training is stopped if performance had not improved for 15 epochs. The best-performing model on the development set is then used for evaluating the test set.

We use the BERT-base uncased model with 12 transformer blocks, a hidden layer size of 768 and 12 attention heads. The embedding vector size is thus 768.

4.2.4. Results

Without transfer: In this experiment we use the target domain labeled data only. This is used as a baseline. In Table 8, we illustrate the base model results of GENE NER and PROTEIN NER. We report the macro-averaged and micro-averaged F1 scores for the WordModel, WordCharacter model, WordCharacterAttention model and the WordCharacterBERT model.

The results are reported on two datasets, BioNLP09 (labeled with the PROTEIN tag in BIO format) and BC2GM (labeled with the GENE tag in BIO format).

As can be seen from the results character based representations and the BERT language model improve the NER.

Impact of BERT model embedding representations: In these experiments we compare the improvement of using the embedding obtained with the BERT language model by integrating it with word and character representations. We compare the obtained embeddings with traditional word2vec and character representations. We train the base model considering the following settings (see Table 9):

- *Word representation (WordModel)*: The model uses only the traditional word2vec embeddings.
- *BERT representation (BERTModel)*: The model uses only the BERT embeddings.

- *BERT with character representation (BERTCharacter)*: The model builds word level representation based on concatenation of the BERT embedding and the character embedding.
- *BERT with character and word representation (WordCharacterBERT)*: The model builds word level representations based on the concatenation of BERT embedding, the word2vec embedding and the character embedding representation.

Table 8. Without transfer: Micro- and macro-averaged F1 scores for NER in BioNLP09 and BC2GM.

Dataset(Entity)	Model	Micro F1	Macro F1
PROTEIN (BioNLP09)	WordModel	0.80	0.81
PROTEIN (BioNLP09)	WordCharacter	0.81	0.82
PROTEIN (BioNLP09)	WordCharacterAttention	0.81	0.82
PROTEIN (BioNLP09)	WordCharacterBERT	0.83	0.85
GENE (BC2GM)	WordModel	0.76	0.77
GENE (BC2GM)	WordCharacter	0.79	0.80
GENE (BC2GM)	WordCharacterAttention	0.80	0.81
GENE (BC2GM)	WordCharacterBERT	0.82	0.84

Table 9. Impact of BERT embedding: Comparison of micro- and macro-averaged F1 scores obtained on the BC2GM dataset.

Dataset(Entity)	Model	Micro F1	Macro F1
GENE (BC2GM)	Word representation	0.76	0.77
GENE (BC2GM)	BERT representation	0.81	0.82
GENE (BC2GM)	BERT with Character representation	0.82	0.83
GENE (BC2GM)	BERT with Character and Word representation	0.82	0.84
GENE (BC2GM)	BIOBERT(trained with Wiki,books and Pubmed)	0.83	0.84

As can be seen from Table 9, word level representations based on the concatenation of the BERT, the word2vec and the character embeddings improve the F1-score of NER on the BC2GM dataset. The results also show that integrating BERT embeddings with the character representation alone (BERTCharacter model) or a combination of BERT embeddings with character and word embeddings (WordCharacterBERT) improves the F1-score compared to only using the BERT embeddings (BERTModel). BioBERT (Lee et al. [16]) yields slightly better F1 scores compared to our BERT combined with character information and BERT combined with character and word information. However, BioBERT is pretrained using the BERT model with huge biomedical corpus, books and Wikipedia texts. In our models we however do not use any pretraining on large biomedical corpora but use only the BC2GM dataset for training and the BERT embeddings for fine-tuning.

Fine-tuning the pre-trained model: In order to understand the impact of the transfer learning model, we use the best pre-trained base models from the source dataset and fine-tune it using the target dataset. The different levels of freezing the parameters are: a. *Lastlayer-Finetuning*: Fine-tuning only last layer; b. *SecondBiLSTM-Finetuning*: Fine-tuning till the second BiLSTM layer.

Analyse transferability of different layers: To identify the influence of the number of labeled examples needed in the target dataset we use different percentages of these when training as indicated in Table 10.

From Table 10, it can be seen that for the WordCharacterAttention model, our transfer learning approach improves the performance compared to the without-transfer setting when dealing with a limited number of labelled training examples. The improvement by transfer learning is substantial when the target labelled training data is small (for example, when training with 25% and 10% of the labelled examples). Also when comparing the TL-A and TL-B transfer learning settings, the TL-A setting performs better. This can be explained by the fact that TL-A shares more model parameters compared to TL-B. The effect of transfer learning is more evident when the target training set size is small, with the improvements diminishing gradually or not seen when the target training set obtains its full size.

Table 10. ***GENE* tag(BC2GM) to *PROTEIN* (BioNLP09)**: Transfer learning results for different levels of freezing at different ratios of target training data. All scores reported are micro-F1 scores.

Setting	Model	Fine-Tuning	100%	75%	50%	25%	10%
Without transfer	WordCharacterAttention		0.82	0.80	0.76	0.69	0.50
With transfer (TL-A)	WordCharacterAttention	Lastlayer-Finetuning	0.81	0.81	0.77	0.73	0.54
With transfer (TL-B)	WordCharacterAttention	SecondBiLSTM-Finetuning	0.80	0.79	0.74	0.70	0.52
With BERTWordChar	WordCharacterBERT	Bert-Finetuning	0.83	0.82	0.80	0.77	0.73
With transfer (TL-C)	WordCharacterBERT	Lastlayer-Finetuning	0.81	0.80	0.78	0.79	0.67
With transfer(TL-D)	WordCharacterBERT	SecondBiLSTM-Finetuning	0.80	0.80	0.77	0.79	0.68

With the WordCharacterBERT model, we have used the BERT embedding from the BERT-base uncased model and concatenated it with the word and character level representations. The model is then fine-tuned on the PROTEINS dataset. This approach improves the F1 score compared to not using any transfer. The largest improvement is observed when only 10% of the target training dataset is used (from 0.50 (Without transfer) to 0.73 (With BERTWordChar)). Similarly with transfer settings TL-C and TL-D, transfer learning is especially beneficial for target datasets with a small number of training labels (for example when using 50%, 25% and 10% of the target labelled examples).

We can thus conclude that the representations learned from the source dataset are effectively transferred and learned by the target dataset. With transfer learning, fewer training examples are sufficient to achieve considerable performance for the sequence labelling task evaluated. The impact of transfer learning is more pronounced when target labelled data is minimum, say 10% or 25%, and improvement diminishes as the percentage of target labelled examples increases. This impact is negligible or not seen when the number of labelled target training data increases. The results show that transfer learning indeed helps in mitigating the problems of the availability of few labelled training data by means of reusing the learned representations.

5. Conclusions

We have proposed a number of deep learning models for named entity recognition that rely on character based models. The models were evaluated with two use cases: entity extraction in financial and in biomedical documents. We have built several variant models that integrate word embeddings, attention mechanisms and the BERT language model. In our work we have especially focused on transfer learning by pretraining a model for a certain NER task and than fine-tuning the learned model for another NER task for which there are few labeled training data. The results have shown the effectiveness of transfer learning, for which different methods for fine-tuning the model were proposed.

Author Contributions: Conceptualization, S.F., J.V.L. and M-F.M.; data curation, S.F. and J.V.L.; methodology, S.F.; investigation, S.F. and J.V.L ; writing—original draft preparation, S.F.; writing—review and editing, S.F., J.V.L. and M-F.M.; visualization, S.F.; supervision, project administration and funding acquisition, M-F.M.

Funding: The research in collaboration with the company Contract.fit was sponsored by VLAIO (Flanders Innovation & Entrepreneurship) under contract number HBC.2017.0264, and another part was financed by the SBO project ACCUMULATE (IWT-SBO-Nr. 150056).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dos Santos, C.N.; Guimarães, V. Boosting named entity recognition with neural character embeddings. In Proceedings of the Fifth Named Entity Workshop (NEWS@ACL 2015), Beijing, China, 31 July 2015; pp. 25–33.
2. Kann, K.; Schütze, H. Single-model encoder-decoder with explicit morphological representation for reinflection. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany, 7–12 August 2016.
3. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
4. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
5. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
6. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-aware neural language models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
7. Cao, K.; Rei, M. A joint model for word embedding and word morphology. *arXiv* **2016**, arXiv:1606.02601.
8. Ling, W.; Luís, T.; Marujo, L.; Astudillo, R.F.; Amir, S.; Dyer, C.; Black, A.W.; Trancoso, I. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv* **2015**, arXiv:1508.02096.
9. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. *arXiv* **2016**, arXiv:1603.01360.
10. Miyamoto, Y.; Cho, K. Gated word-character recurrent language model. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, TX, USA, 1–4 November 2016; pp. 1992–1997.
11. Ando, R.K.; Zhang, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.* **2005**, *6*, 1817–1853.
12. Peng, N.; Dredze, M. Multi-task multi-domain representation learning for sequence tagging. *arXiv* **2016**, arXiv:1608.02689.
13. Yang, Z.; Salakhutdinov, R.; Cohen, W.W. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv* **2017**, arXiv:1703.06345.
14. Rodriguez, J.D.; Caldwell, A.; Liu, A. Transfer learning for entity recognition of novel classes. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 1974–1985.
15. Lee, J.Y.; Dernoncourt, F.; Szolovits, P. Transfer learning for named-entity recognition with neural networks. *arXiv* **2017**, arXiv:1705.06273.
16. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *arXiv* **2019**, arXiv:1901.08746.
17. Lafferty, J.D.; McCallum, A.; Pereira, F.C.N. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML), Williamstown, MA, USA, 28 June–1 July 2001; pp. 282–289.
18. Sakharov, A.; Sakharov, T. The Viterbi algorithm for subsets of stochastic context-free languages. *Inf. Process. Lett.* **2018**, *135*, 68–72. [CrossRef]
19. Rei, M.; Crichton, G.K.O.; Pyysalo, S. Attending to characters in neural sequence labeling models. In Proceedings of the 26th International Conference on Computational Linguistics (COLING), Osaka, Japan, 11–16 December 2016; pp. 309–318.
20. Howard, J.; Ruder, S. Universal Language Model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL, Melbourne, Australia, 15–20 July 2018; pp. 328–339.
21. Erhan, D.; Manzagol, P.; Bengio, Y.; Bengio, S.; Vincent, P. The difficulty of training deep architectures and the effect of unsupervised pre-training. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS), Clearwater, FL, USA, 16–18 April 2009; pp. 153–160.

22. Giorgi, J.M.; Bader, G.D. Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics* **2018**, *34*, 4087–4094. [CrossRef] [PubMed]
23. Smith, L.; Tanabe, L.K.; nee Ando, R.J.; Kuo, C.J.; Chung, I.F.; Hsu, C.N.; Lin, Y.S.; Klinger, R.; Friedrich, C.M.; Ganchev, K.; et al. Overview of BioCreative II gene mention recognition. *Genome Biol.* **2008**, *9*, S2. [CrossRef] [PubMed]
24. Kim, J.D.; Ohta, T.; Pyysalo, S.; Kano, Y.; Tsujii, J. Extracting bio-molecular events from literature: The BIONLP'09 shared task. *Comput. Intell.* **2011**, *27*, 513–540. [CrossRef]
25. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.