Master Thesis

Job Salary Prediction using Neural Language Models, Conformalized Quantile Regression and Transfer Learning

Anthony Yazdani
Author
Anthony.Yazdani@etu.unige.ch

Prof. Sebastian Engelke
Supervisor
Sebastian.Engelke@unige.ch

University of Geneva
Geneva School of Economics and Management
Master of Science in Statistics

Accademic Year 2020-2021

Abstract

The objective of this master's thesis is to use natural language processing models to predict an individual's expected salary based on the job description for which he or she is applying. This master thesis presents two particularly effective models for text-related tasks, that is, BLSTM and BERT. In addition, we put into practice a novel method for conformal inference, the conformalized quantile regression. We aim at developing models for the Swiss labor market. To this end, we considered using transfer learning to take advantage of larger datasets. We also believe that lightening complex models is essential for fast training and easy industrialization. Therefore, we also expose our method and the results obtained using weight sharing techniques.

Keywords: neural language models, BLSTM, BERT, conformal inference, conformalized quantile regression, transfer learning, weight sharing.

Acknowledgements

First of all, I would like to thank my family, who have always supported me and given me the opportunity to study at the university. I would also like to thank Professor Engelke and Olivier Pasche for their help and the opportunity they gave me to study this subject. I would like to thank the University of Geneva for all that I have learned there. Of course, I would also like to thank Mehdi Hirari, without whom this adventure at the University of Geneva would not have been as enriching.

Contents

1 Introduction **5**

2 Literature Overview **6**
2.1 Numerical Representation of Natural Language 6
2.1.1 Word2Vec 7
2.1.2 GloVe 9
2.2 Regression Theory and Conformal Prediction Intervals 11
2.3 Models 12
2.3.1 Multilayer Perceptron 13
2.3.2 BLSTM 13
2.3.3 BERT 15
2.4 Double Transfer Learning 22
2.5 Weight Sharing 22

3 Experimental Setting **23**

4 Results **25**
4.1 Models on the U.S. Dataset D_1 25
4.2 Models on the Swiss Dataset D_3 30

5 Discussion and Conclusion **34**

A Appendix **36**
A.1 Pinball loss 36
A.2 Example of job description 37
A.3 Exploratory Data Analysis 39

1 Introduction

Natural Language Processing (NLP) is a branch of Machine Learning (ML) which focuses on enabling computers to comprehend and manipulate human language. NLP has emerged in many fields, including computer science and statistics, to improve human-machine communication and solve complex tasks involving textual data. This includes machine translation, text summarization or text classification. With the progress of researchers, NLP models have become more and more powerful. Recently, the Transformer model has completely changed the field of machine translation by achieving state-of-the-art results (Vaswani et al., 2017). One of the peculiarities of the NLP field is that the models can be easily modified to fit other tasks. In fact, the BERT model based on the Transformer architecture shows excellent performance, especially in question answering tasks such as SQuAD v1.1 and SQuAD v2.0 (Devlin et al., 2018; Rajpurkar et al., 2016, 2018). Beyond the scientific advances in the field, NLP is becoming more and more important due to the substantial increase in computing power among individuals and small companies, allowing them to solve text-related problems that they were unable to automate before. All businesses can now have their own powerful chatbots or quantify the urgency of unread customer requests easily.

The purpose of this master's thesis is to use NLP models to predict a person's expected salary based on the description of the position they are applying for. In this project, we assume that the salary is mainly explained by the responsibilities, tasks and skills written in the description. We aim to develop a tool that can serve both job seekers and companies. Indeed, it can be difficult for a company to estimate the salary of a position because it may be rare or have no equivalent in the labor market. For candidates, whether they are looking for a job or a promotion, we believe it is important to know what salary one aspires to, in order to negotiate a fair compensation.

As we want to develop models and analyses for the Swiss labor market in particular, we face important limitations in terms of available data. We decided to use transfer learning in order to address this shortcoming. We aim at studying whether we can benefit from pre-trained models on larger datasets before fine-tuning these models for our main task. In this master's thesis, we first propose to use a special type of Recurrent Neural Networks (RNN), namely, a BLSTM (Hochreiter and Schmidhuber, 1997) and what we call double transfer learning. We also propose to use a BERT pre-trained model (Devlin et al., 2018), which we want to fine-tune for our own task.

Furthermore, we argue that generating a point estimate is not sufficiently informative. One of the goals of this master's thesis is to create models that can represent uncertainty in the prediction of future observations. We believe it is critical to provide an interval for predictions, which can reflect either the volatility of a job's wage or the imprecision of the description. To do so, we rely on the method proposed by Romano et al. (2019), which they call Conformalized Quantile Regression (CQR).

Since NLP models tend to be numerically expensive, we are also exploring the idea of weight sharing. Indeed, many architectures base their results on the number of parameters, as shown in GPT-3 (Brown et al., 2020) (175 billion parameters) or Wu Dao 2.0 (1750 billion parameters). It is very difficult for individuals or small businesses to benefit from these new technologies because they are expensive to train and require a lot of hardware to make them operational. For example, GPT-3 cost OpenAI $12 million to train. Beyond the cost, these models are also extremely difficult to train from an optimization point of view. We therefore argue that it is necessary to reduce the size of the models to make them more accessible, fast to train and easy to industrialize. Many works go in this direction, for example the Fnet architecture (Lee-Thorp et al., 2021). To this end, we develop models that provide both prediction intervals and expected salaries, minimizing a linear combination of pinball and root mean squared error loss functions. Besides the numerical complexity, we also believe that these different tasks can share the same explanatory variables and that it is not necessary to train a distinct model for each task.

2 Literature Overview

2.1 Numerical Representation of Natural Language

To accomplish our tasks, we first need to represent our text data in a form that can be manipulated by machine learning models. To do this, we follow a two-step procedure: text segmentation and word embedding. Text segmentation, or tokenization, is the process of breaking down text into meaningful parts. Therefore, we can represent a textual instance as a sequence of words.

$$[\text{Fun with machine learning.}] \longrightarrow [[\text{Fun}], [\text{with}], [\text{machine}], [\text{learning}], [.]]$$

Word tokenization in its most basic form is not the only viable method. Wu et al. (2016), for example, divide text into a small collection of common sub-words, a technique known as wordpiece. This method naturally handles rare words, because it avoids having to deal with all the sub-word combinations. This method has proven to be effective, especially in the BERT model.

$$[\text{Fun with machine learning.}] \longrightarrow [[\text{Fun}], [\text{with}], [\text{machine}], [\text{learn}], [\#\text{ing}] [.]]$$

Each of these tokens can then be vectorized using word embedding methods. In this master's thesis, we will use two well-known methods, namely Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Word embedding, or word vectorization, is a concept that allows to represent each word in a corpus as a vector of real numbers. In this new representation, terms that appear in a similar way have corresponding vectors that are relatively close. This strategy is based on the distributional hypothesis (Sahlgren, 2008), which states that words with similar meanings appear in similar contexts.

2.1.1 Word2Vec

The Continuous Bag of Words (CBOW) and the Skip Gram (SG) models are the two variants of Word2Vec. We will focus on SG, which is the model that was emphasized in the original article. This model was created with the aim of predicting context words based on a center word. A detailed explanation can be found in (Rong, 2014), including the model formulation and the backpropagation equations. In a nutshell, this is how the model is defined.

Let o_i be a one hot-encoded vector of size V at the i^{th} coordinate, where V is the size of our vocabulary (the number of unique words in our corpus). Let E be a $V \times N$ matrix of randomly initialized word embeddings, where N is a hyper-parameter to be set. Furthermore, let C be a $V \times N$ matrix of randomly initialized context embeddings. Therefore, the i^{th} word (w_i) in our vocabulary has both a corresponding embedding vector E_i and a context vector C_i. A context word C_j is defined as a positive context for a word E_i if it appears within p position of E_i in our corpus. The parameter p is also a hyper-parameter referred to as the window size. For a word i, positive context words are labeled as $y_{ij} = 1$ while negative context words are labeled as $y_{ij} = 0$. The SG model can be represented as a neural network, with the following forward pass.

We select the embedding of the i^{th} word (w_i)

$$E_i = o_i \cdot E, \tag{1}$$

we compute the scores with context vectors

$$s_i = E_i \cdot C^T, \tag{2}$$

we compute the probabilities of observing the context words w_j given the word of interest w_i

$$P(w_j|w_i) = \text{Softmax}\left(s_{ij}\right), \tag{3}$$

and we compute the negative log likelihood to be minimized

$$\begin{aligned}
\mathcal{L} &= -\ln\left(\prod_{j=1}^{V} P(w_j|w_i)^{y_{ij}}\right) \\
&= -\sum_{j=1}^{V} y_{ij}\ \ln\left(\frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{V}\exp\left(s_{ik}\right)}\right) \\
&= (2p)\times\ln\left(\sum_{k=1}^{V}\exp\left(s_{ik}\right)\right) - \sum_{j=1}^{V} y_{ij}\ s_{ij}. \qquad (4)
\end{aligned}$$

Finally, E_i and C are updated by a gradient-based optimization algorithm. Once the optimization is complete, we can use the vectors of the matrix E to mathematically represent the words of our descriptions.

One can notice that if a word is interchangeable with another, then their embeddings must be identical, which satisfies the distributional hypothesis. One can also note that this algorithm can be computationally expensive if V becomes large, which is the usual scenario in NLP tasks. Negative sampling is the simplest approach to overcome this shortcoming (Goldberg and Levy, 2014). Negative sampling is essentially a modification of the previous algorithm so that we do not sum over all the negative context words in equation (4).

The quality of word vectors increases significantly with the amount of training data. To take advantage of this, we chose to use Google's pre-trained vectors. These embeddings were trained on 100 billion words from the Google News dataset. The final model provides 3 million unique words embedded in 300 dimensional vectors.

2.1.2 GloVe

GloVe, for Global Vectors, is also an unsupervised learning algorithm for obtaining vector representations of words. The learning is based on the windowed co-occurrence matrix of words in a corpus. It turns out to be a particular form of matrix factorization.

The first step is to construct a windowed co-occurrence matrix, denoted Cooc of dimension $V \times V$. An element of Cooc, that is Cooc_{ij}, represents the number of times the word w_i was not further than p position of the word w_j in the whole corpus. If our corpus was [Fun with machine learning.] and $p = 2$, our co-occurrence matrix would be

	Fun	with	machine	learning
Fun	1	1	0	0
with	1	1	1	0
machine	0	1	1	1
learning	0	0	1	1

The essential next step is to decompose the Cooc matrix. In fact, the GloVe loss function turns out to be a weighted log-regression model such that

$$\min_{E^{(1)},E^{(2)},b} \sum_{i,j=1}^{V} f\left(\text{Cooc}_{\text{ij}}\right) \left(E_i^{(1)} {E_j^{(2)}}^T + b_i + b_j - \ln\left(\text{Cooc}_{\text{ij}}\right)\right)^2, \tag{5}$$

where

$$f(x) = \begin{cases} (x/100)^{3/4} & \text{if } x < 100 \\ 1 & \text{otherwise.} \end{cases}$$

The parameters are updated by a gradient-based optimization algorithm, and the final embedding matrix E is computed as $E = E^{(1)} + E^{(2)}$.

The weighting function $f(x)$ shown in Figure 1 satisfies some convenient properties. Indeed, frequent and rare words will not contribute excessively to the embeddings. For example, we do not want the words "the" or "and" to contribute much because they do not allow us to learn the semantic relationship between words. Similarly, words that rarely appear together are assumed to be noisy and should not be considered.

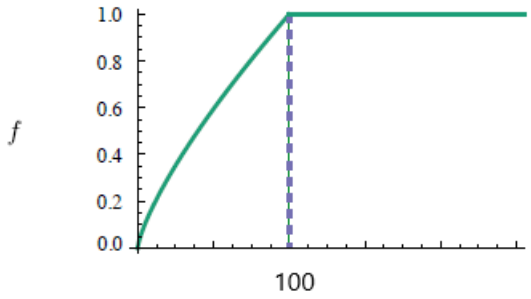


Figure 1: Weighting function f as defined in the GloVe loss function. Source: (Pennington et al., 2014)

There are several possibilities to deal with $\ln(\text{Cooc}_{ij})$ in equation (5) when Cooc_{ij} is equal to zero. We can either add 1 to all the values of the matrix, or run the algorithm over the non-zero values of the matrix. The second choice is often preferred because it reduces the computational complexity.

Similar to Word2Vec, the quality of word vectors also increases significantly with the amount of training data. Therefore, we decided to use Stanford's pre-trained vectors. These embeddings were trained on 840 billion words from the Common Crawl dataset. The final model provides 2.2 million unique words embedded in 300 dimensional vectors.

2.2 Regression Theory and Conformal Prediction Intervals

As we want to use CQR, we begin by splitting the data into a proper training set $\{(X_i, Y_i) : i \in \mathcal{I}_1\}$, a calibration set $\{(X_i, Y_i) : i \in \mathcal{I}_2\}$ and a validation set $\{(X_i, Y_i) : i \in \mathcal{I}_3\}$.

For classical regression, we fit a function $f_{\mathbf{w}}(X_i)$ for $i \in \mathcal{I}_1$, by choosing the parameter $\mathbf{w}$ minimizing the Root Mean Squared Error (RMSE)

$$\mathcal{L}(\mathbf{w}) := \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - f_{\mathbf{w}}(X_i)\right)^2}. \tag{6}$$

Additionally, we can construct a prediction interval $C(X_i)$ for $i \in \mathcal{I}_3$ such that

$$\mathbb{P}\{Y \in C(X) \mid X = x\} \geq 1 - \alpha. \tag{7}$$

To do so, one could use the absolute residuals computed on the calibration set

$$R_i := \left|Y_i - f_{\mathbf{w}}(X_i)\right|, \quad i \in \mathcal{I}_2. \tag{8}$$

We then could compute a quantile of the empirical distribution of the absolute residuals

$$Q_{1-\alpha}(R, \mathcal{I}_2) := (1-\alpha)\left(1 + 1/|\mathcal{I}_2|\right)\text{-th quantile of}\left\{R_i : i \in \mathcal{I}_2\right\}, \tag{9}$$

and output the prediction interval for a validation point X_i as

$$C(X_i) := \left[f_{\mathbf{w}}(X_i) - Q_{1-\alpha}(R, \mathcal{I}_2), f_{\mathbf{w}}(X_i) + Q_{1-\alpha}(R, \mathcal{I}_2)\right]. \tag{10}$$

However, we want the prediction intervals to vary with the job description. To do this, we follow the method proposed by Romano et al. (2019). This method is a slight modification of the one just described. We proceed analogously to (6). We first fit two quantile functions, $q_{(\alpha_l,\mathbf{w_l})}(X_i)$ and $q_{(\alpha_u,\mathbf{w_u})}(X_i)$ for $i \in \mathcal{I}_1$ by minimizing the pinball loss function defined as

$$\mathcal{P}(\theta, \mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} \left(\theta - \mathbb{1}\{e_i < 0\}\right) e_i, \tag{11}$$

where θ is the level of interest and $e_i = Y_i - q_{(\theta,\mathbf{w})}(X_i)$.

The proof that this lead to a θ-quantile regression is detailed in A.1. Finally, we correct the errors made by the base prediction interval for a validation point as follows

$$C(X_i) := \left[q_{(\alpha_l,\mathbf{w_l})}(X_i) - Q_{1-\alpha_l}\left(E^l\right) \ , \ q_{(\alpha_u,\mathbf{w_u})}(X_i) + Q_{1-\alpha_u}(E^u)\right], \tag{12}$$

where

$$Q_{1-\theta}(A) := (1-\theta)\left(1 + 1/|\mathcal{I}_2|\right) \text{ -th empirical quantile of } A, \tag{13}$$

$$E_i^l := q_{(\alpha_l,\mathbf{w_l})}(X_i) - Y_i, \quad i \in \mathcal{I}_2, \tag{14}$$

$$E_i^u := Y_i - q_{(\alpha_u,\mathbf{w_u})}(X_i), \quad i \in \mathcal{I}_2. \tag{15}$$

2.3 Models

As discussed in Section 2.2, we need to find three functions $f_{\mathbf{w}}(X_i)$, $q_{(\alpha_l,\mathbf{w_l})}(X_i)$ and $q_{(\alpha_u,\mathbf{w_u})}(X_i)$ to minimize the loss functions, so as to produce close-to-reality estimates and shortest possible prediction intervals. The universal approximation theorem states that a simple feed-forward neural network can approximate any continuous function under certain conditions (Cybenko, 1989; Leshno et al., 1993). In practice, we need more flexible and powerful upstream models to interpret textual data. We choose to use two models that we believe are most suitable for processing natural language, namely the Bi-directional Long-Short Term Memory neural network (BLSTM) and the Bi-directional Encoder Representations from Transformers (BERT). Regardless of whether the data is processed by one or the other, the resulting data representation is then fed to a MultiLayer Perceptron (MLP). In this section, we will briefly recall what an MLP is, and then we will elaborate on both BLSTM and BERT models.

2.3.1 Multilayer Perceptron

The MLP is an artificial neural network composed of several layers in which information flows only from the input layer to the output layer. Such a network is called a feed-forward network. Mathematically, the output $x^{(l)}$ of the l^{th} layer is a function of the output of the previous layer, i.e.,

$$\mathbf{x}^{(l)} = h^{(l)}\left(\mathbf{x}^{(l-1)}\right) = \phi\left(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{16}$$

where ϕ is the activation function, $\mathbf{W}^{(l)}$ is the l^{th} weight matrix and $\mathbf{b}^{(l)}$ is the l^{th} bias vector.

The final function that is applied, denoted $h^{(L)}$, depends on the underlying task. It can be adjusted to do classification or regression. For our regression tasks, $h^{(L)}\left(\mathbf{x}^{(L-1)}\right) = w^{(L)}\mathbf{x}^{(L-1)} + b^{(L)}$, where $w^{(L)}$ is a row vector and $b^{(L)}$ is a scalar. Figure 2 illustrates an example of such a network with two hidden layers.

2.3.2 BLSTM

A Recurrent Neural Network (RNN) is a type of neural network in which the connections between nodes form a time-directed graph. RNNs are non-linear auto-regressive models. Therefore, they can handle complex temporal behaviors. Their use in text related tasks comes both from the assumption that natural language is sequential and from their capacity to handle sequences of varying dimensions. RNNs are therefore well suited in this context because we can represent a text into sequence of tokens (see Section 2.1). However, RNNs can be subject to two problems, vanishing gradient or gradient explosion (Bengio et al., 1994). There are several techniques that can be used to avoid these phenomena. For example, we can truncate the backpropagation at a maximum time lag (Williams and Peng, 1990) or we can use gradient clipping (Pascanu et al., 2013). Truncated backpropagation is useful for limiting the vanishing gradient effect, however, it has no impact on the learning ability of RNNs because the signal from truncated time steps is not taken into account. An effective approach is to ensure that relevant information from previous temporal steps is retained in recurrent cells near the output. To do this, we use gating mechanisms.

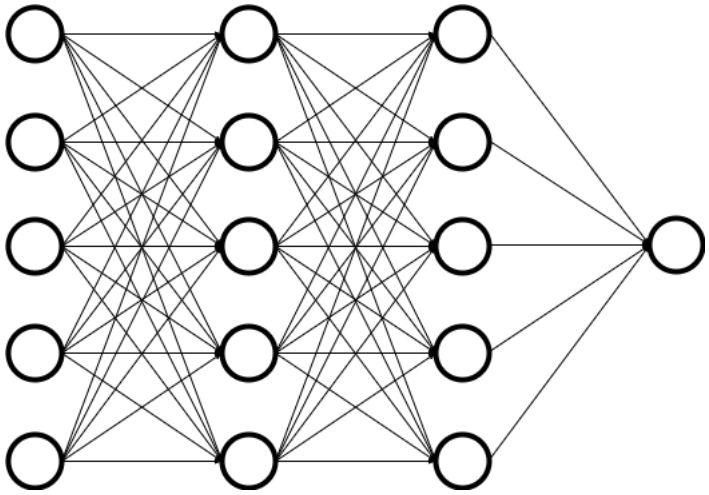


Figure 2: Graphical representation of a MultiLayer Perceptron (MLP) with two hidden layers.

The LSTM model includes a gating mechanism for long-term memory storage. An LSTM cell is composed of the following elements

$$\Gamma_{ui} = \sigma\left(U_u\ a_{i-1} + W_u\ E_i + b_u\right), \tag{17}$$

$$\Gamma_{fi} = \sigma\left(U_f\ a_{i-1} + W_f\ E_i + b_f\right), \tag{18}$$

$$\Gamma_{oi} = \sigma\left(U_o\ a_{i-1} + W_o\ E_i + b_o\right), \tag{19}$$

$$\tilde{C}_i = tanh\left(U_g\ a_{i-1} + W_g\ E_i + b_g\right), \tag{20}$$

$$c_i = \Gamma_{fi} \odot c_{i-1} + \Gamma_{ui} \odot \tilde{C}_i, \tag{21}$$

$$a_i = \Gamma_{oi} \odot \tanh\left(c_i\right), \tag{22}$$

where Γ_{ui} is the update gate, Γ_{fi} is the forget gate, Γ_{oi} is the output gate, $\tilde{C}_i$ is the cell state candidate, c_i is the cell state, a_i is the hidden state, $\sigma(x) = \frac{1}{1+e^{-x}}$ and $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The output of the LSTM cell is a function of E_i, the embedding of the i^{th} word of the sequence, and a_{i-1}, the output from the previous cell. The initial vector a_0 is initialized as the zero vector.

Knowing that applying $\sigma(.)$ produces a result between 0 and 1, the purpose of gates is to learn what to update, forget or output through the Hadamard product. The cell state c_i, removes useless information from past memory with the forget gate and updates new information in memory with the update gate. Finally, the hidden state a_i is a nonlinear function of the cell state.

LSTMs also have limitations, as future information cannot be obtained from the current hidden state. A common practice is to assemble two independent LSTMs. This structure allows the networks to have information from left-to-right and right-to-left context. The final step is to concatenate the last hidden states, and add a regression layer on top of this new data representation. Such a model is called a Bi-directional LSTM, or BLSTM. Figure 3 shows our BLSTM architecture in more detail.

2.3.3 BERT

Since natural language is complex and diverse, models generally require many observations to generalize properly. Therefore, NLP researchers have been looking for ways to pre-train models using large amounts of unlabeled data. Devlin et al. (2018) have proposed the BERT model.

In addition to being pre-trained on a massive amount of data, BERT produces deep bi-directional representations, i.e., is able to change the representation of a word depending on its context. GloVe, for example, generates a single word embedding for each word in the vocabulary. BERT, on the other hand, takes into account the context of each word to produce an embedding. For example, the word "bank" will not have the same representation when it appears in the sentences "bank of the river" and "a bank account". The success of this model is derived from its architecture which is based on the Transformer, first introduced by Vaswani et al. (2017). Figure 4 shows a graphical representation of the BERT model.

In this section, we will first explain the self-attention mechanism and then the model as a whole. In a second step, we will discuss the way this model has been pre-trained and how we can fine-tune it for our own task.

Self Attention Mechanism

As mentioned earlier, BERT derives its effectiveness in part from the contextualization of words. This is achieved by stacking Transformer blocks. A Transformer block is composed of several elements of which the heart is the Multi-Head Attention layer (see Figure 5).

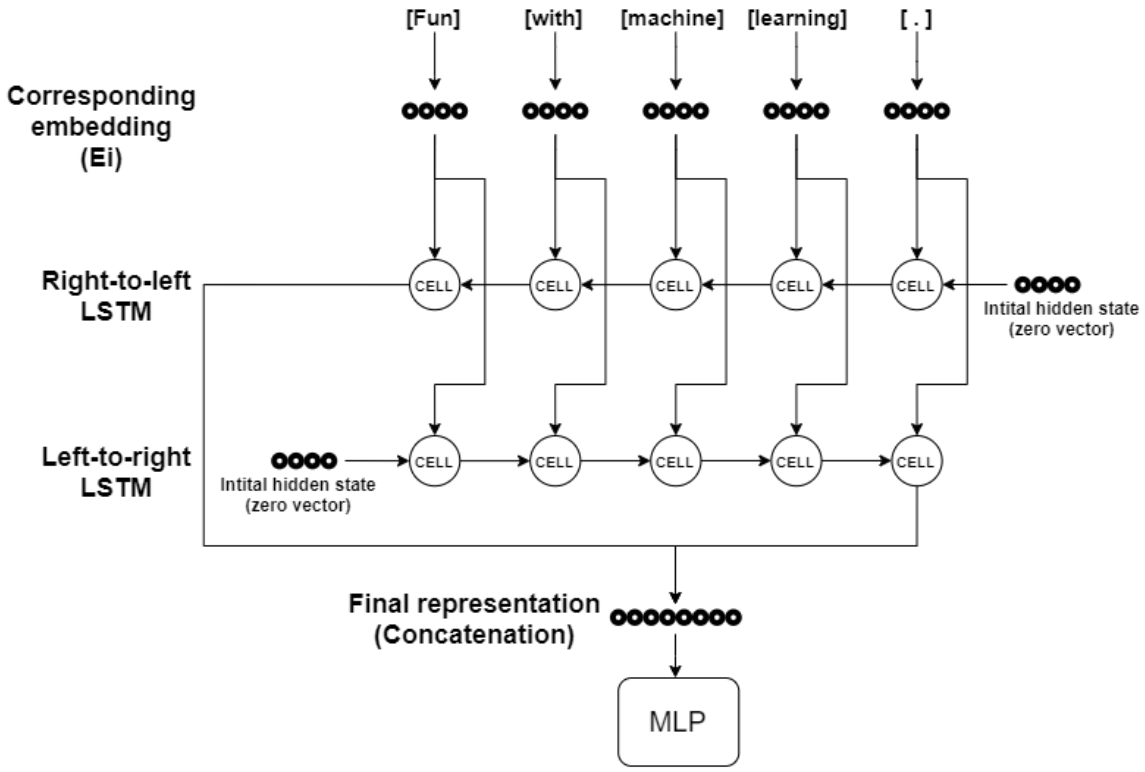


Figure 3: Bi-directional Long-Short Term Memory neural network (BLSTM) computational graph for text regression task.

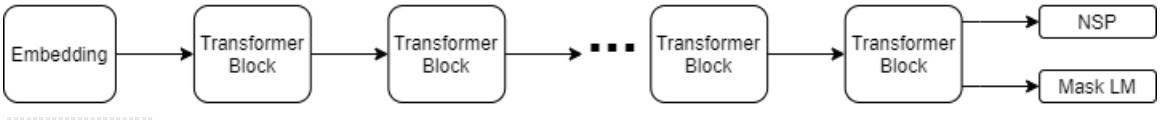


Figure 4: Bi-directional Encoder Representations from Transformers (BERT) computational graph.

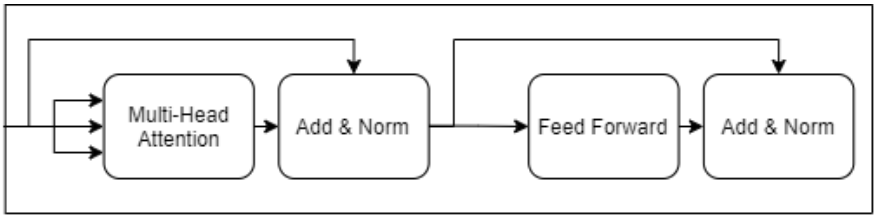


Figure 5: Detailed architecture of the Transformer block in the BERT model.

Formally, the self-attention can be described as a simple mechanism which involves basic algebraic operations on query, key and value matrices. Let us introduce the key matrix M_K, the query matrix M_Q and the value matrix M_V and compute the transformation of the embedding matrix $\tilde{E}$ as

$$Q = \tilde{E} \cdot M_Q, \tag{23}$$

$$K = \tilde{E} \cdot M_K, \tag{24}$$

$$V = \tilde{E} \cdot M_V. \tag{25}$$

The self-attention is obtained as follows

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, \tag{26}$$

where d_k is the queries and keys dimension and act as a scaling factor for numerical stability. When the dimensionality is large, the QK^T operation produces large values, pushing the gradients to be exceedingly small. For the base version of BERT, we have that $d_k = 768$, whereas for its large version we have $d_k = 1024$.

Multi-Head Attention Mechanism

The term "Multi-Head" means that the attention mechanism occurs multiple times in a single Transformer block. This allows us to leverage information from different sub-spaces. The self-attention function runs in parallel on h different query, key and value matrices. This therefore produces h different output matrices, called "heads". We have M_K^i, M_Q^i, and M_V^i for $i = \{1, \ldots, h\}$. Once the h different new embeddings are produced, they are concatenated and projected onto a space of the same dimension as $\tilde{E}$ through a matrix W_O.

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W_O, \tag{27}$$

where $\text{head}_i = \text{Attention}\left(\tilde{E} \cdot M_Q^i \ , \ \tilde{E} \cdot M_K^i \ , \ \tilde{E} \cdot M_V^i\right)$.

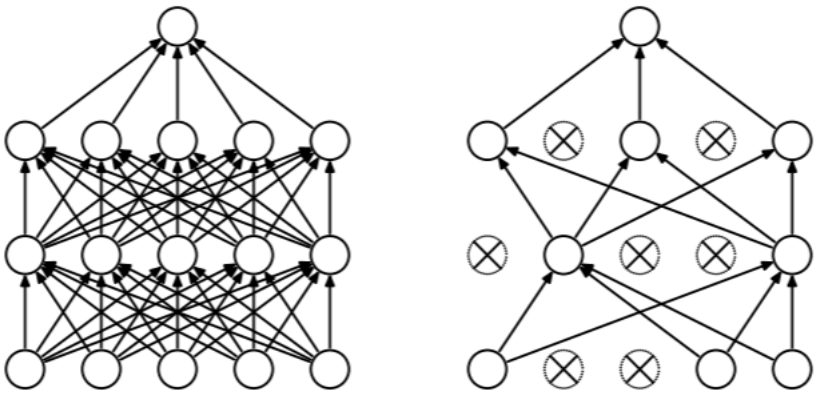
The number of heads varies depending on the version of BERT used. For the base BERT model we have 12 heads and for the large BERT we have 16 heads.

Model Architecture

The BERT model is mostly composed of a stack of Transformer blocks, as shown in Figure 4. The BERT base has 12 blocks while the large version has 24 blocks. Each block has a multi-head attention mechanism, skip connections, layer normalization (Ba et al., 2016) and feed-forward layers. Skip connections allow to skip certain layers of the neural network and use the output of one layer as an additional input to subsequent layers. Roughly speaking, this gives an alternative path to the gradient during back-propagation. The layer normalization has also been introduced to facilitate the learning process. Indeed, Ioffe and Szegedy (2015) have empirically proven that normalization reduces Internal Covariate Shift (ICS), a phenomenon defined as the change in the distribution of network activations due to changes in parameters during training. They show that ICS is detrimental to achieving fast convergence and describe a variety of additional properties of normalization. These include prevention of gradient explosion or vanishing, as well as robustness to various hyper-parameter settings such as the learning rate. Moreover, dropout (Srivastava et al., 2014) is used as a regularization method in feed-forward layers of each Transformer block. When dropout is used, some units in hidden layers are dropped with probability p at each batch processing, and the parameters are updated without them. By doing so, we can theoretically obtain up to $2^{\#\text{units}}$ various weight-shared architectures (thinned networks). All hidden units are used in the testing phase, but their weights are multiplied by p so that the output matches approximately the expected output of all thinned networks. For example, equation (16) would become

$$\begin{aligned} \mathbf{x}^{(l)} &= h^{(l)}\left(\mathbf{x}^{(l-1)} \odot \mathbf{r}^{(l-1)}\right) \\ &= h^{(l)}\left(\mathbf{z}^{(l-1)}\right) \\ &= \phi\left(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}\right) \end{aligned} \tag{28}$$

where $\mathbf{r}^{(l-1)} \sim \text{Bernoulli}(p)$. Figure 6 shows an illustration of dropout regularization.



(a) Standard Neural Net (b) After applying dropout.

Figure 6: Graphical representation of the dropout regularization. Each crossed-out node in (b) represents a dropped node. Source: (Srivastava et al., 2014)

The embedding layer

The embedding layer is different from the one we saw in the BLSTM model. Word order is taken into account by construction in RNNs. However, with the Transformer block, all words are processed at the same time, so our model loses the notion of order. This is why positional embedding is introduced in the model. The importance of word order can be seen by examining these two sentences: "Even though she did not win the award, she was satisfied" and "Even though she did win the award, she was not satisfied".

The idea is to add a matrix P of dimension $dim(E)$ to the embedding matrix E, which is meant to represent the order of the tokens in the sequence.

$$\tilde{E} = P + E. \tag{29}$$

Note that this model uses the wordpiece tokenization method that we discussed in Section 2.1.

Pre-Training and Fine-Tuning

The BERT model was pre-trained on two tasks simultaneously. The first task is to find missing words in a sentence, a task called Mask Language Model (Mask LM). The second task is called Next Sentence Prediction (NSP). The model is given two sentences, and its objective is to determine whether the second sentence provided is indeed a continuation of the first. The team responsible for BERT trained this model on the BooksCorpus dataset (Zhu et al., 2015) (800 million words) and the entire English Wikipedia (2500 million words).

For the Mask LM task, given an input sequence, we first select a random set of q tokens to mask. BERT then learns to predict the original words, minimizing the cross-entropy between the predicted words and the true words. Figure 7 illustrates the forward pass for this task. The cross entropy is computed on all V tokens in our vocabulary and then averaged on the number of masked words.

$$\mathcal{MLM} := \frac{1}{q} \sum_{i=1}^{q} \text{CrossEntropy}\left(P_i \ , \ \text{TrueLabels}_\text{i}\right). \tag{30}$$

For the NSP task, the model is given two sentences and trained to predict whether the second sentence should be placed after the first by minimizing a logistic loss, as shown in Figure 8.

The idea of this model is now to take everything but the last layer and train it on our own tasks through the original input for the NSP layer (see Figure 9).

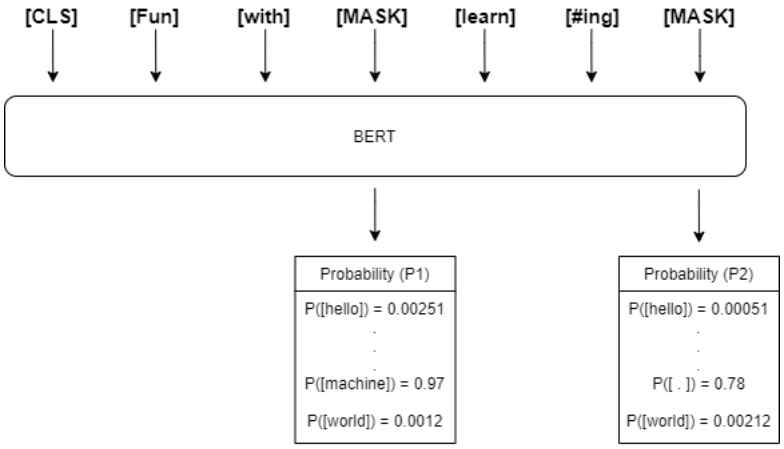


Figure 7: Graphical representation of the first task of the BERT model pre-training. It consists in finding the missing words in a sentence, a task called Mask Language Model.

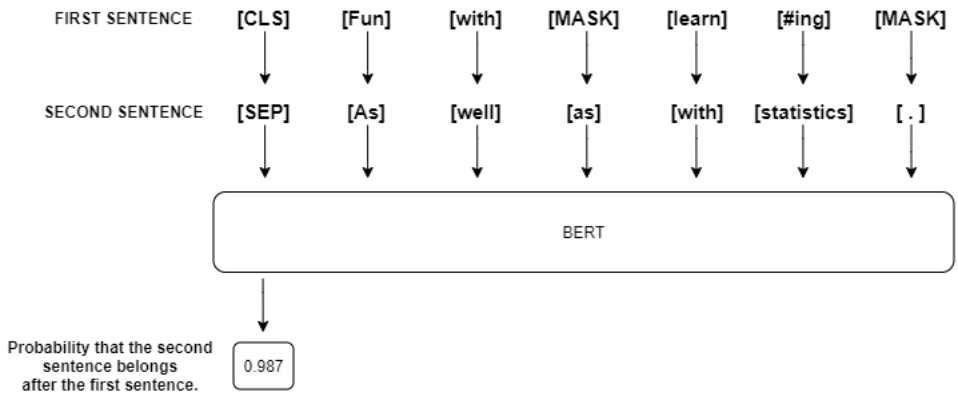


Figure 8: Graphical representation of the second task of the BERT model pre-training. Given two sentences, it consists in determining if the second sentence provided is a continuation of the first one.

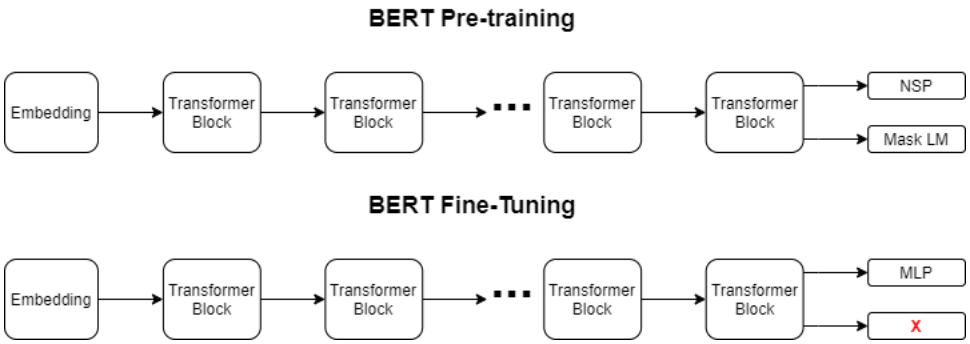


Figure 9: The two fundamental steps to use a BERT model for a downstream task, that is, the pre-training as well as fine tuning.

2.4 Double Transfer Learning

Transfer learning aims to transfer knowledge from a source model to one or more target tasks. Many effective pre-training approaches are based on language modeling (LM), such as BERT. However, we believe that applying transfer learning in the context of supervised models may be effective. Although methods like Word2Vec or GloVe already provide good word representations, we believe that supervised pre-training will allow for better generalization. Indeed, we think that pre-training on a larger dataset would allow the model to be enriched with a substantial amount of new vocabulary and to refine the word vectors to better match our tasks. We plan to implement this supervised transfer learning with BLSTM models. More precisely, we want to implement what we call double transfer learning. That is, we pre-train a BLSTM model on a first dataset (D_1), then we fine-tune it for a second dataset (D_2) and finally we fine-tune it on our dataset of interest (D_3). Details on these datasets can be found in Section 3.

2.5 Weight Sharing

Weight sharing (WS) is a technique to reduce the number of weights in a neural network. In our example, it consists in reusing some parameters for different tasks. A typical application of weight sharing is the Convolutional Neural Network (CNN), in which we share the kernel weights, which significantly reduces the learning time. In addition, our tasks are likely to share common features useful for prediction. If so, we could achieve similar performance using three distinct functions or a model that shares most of its weights.

Our framework allows us to fit independent functions for our three regression models, as shown in Figure 10. But we can also share some parameters. To do so, we can define a third loss function as a linear combination of pinball and RMSE loss functions

$$\Psi(\alpha_l, \alpha_u, \mathbf{W}) := \kappa \times \mathcal{L}(\mathbf{W}) + \lambda \times \mathcal{P}(\alpha_l, \mathbf{W}) + \mu \times \mathcal{P}(\alpha_u, \mathbf{W}), \tag{31}$$

where κ, λ and μ are three real valued scalars.

The parameters to be shared naturally follow from the architecture of the models we use. The most convenient method is to share the parameters of the upstream models (as defined in Section 2.3.2 and 2.3.3) and add an MLP on top of these new data representations. Figure 11 shows the complete pipeline from job descriptions to predictions with weight sharing.

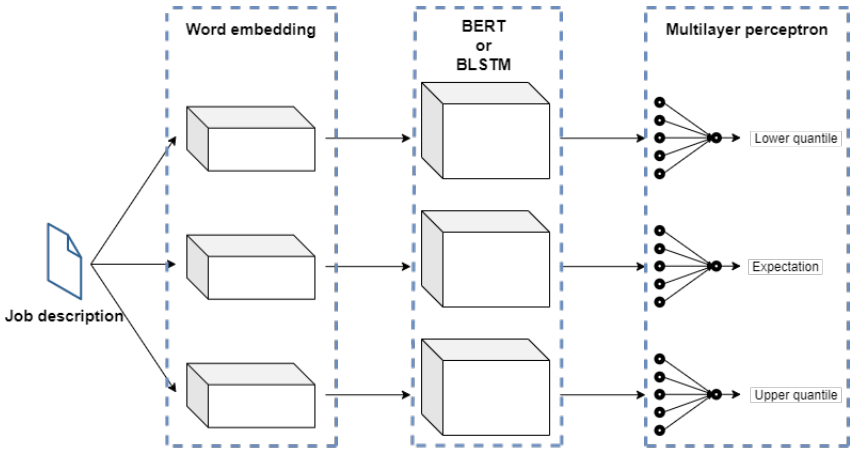


Figure 10: Model pipeline from job descriptions to quantities of interest without weight sharing.

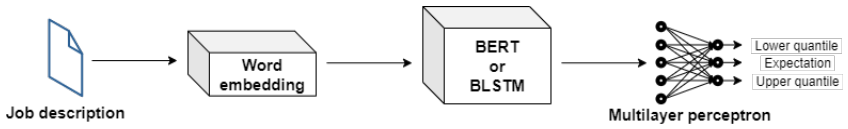


Figure 11: Model pipeline from job descriptions to quantities of interest with weight sharing.

3 Experimental Setting

We will use three datasets, called D_3, D_2 and D_1. These datasets come from two different sources. D_1 comes from salary.com (Plunkett, 1999), and D_2 and D_3 come from payscale's website (Giordano and Gaffney, 2002). Our main goal is to develop models for the D_3 dataset, as this is the one that corresponds to the Swiss labor market. The other two are observations from the U.S. labor market. Each of these datasets consists of job descriptions (X_i) (e.g. A.2) and their corresponding median salary (Y_i). The descriptions associated with these jobs are generic descriptions that describe a job in a global way. D_3, D_2 and D_1 are approximately composed of 1000, 6000 and 33000 instances respectively. We want to take advantage of D_1 as we believe that its large number of observations will enable us to train suitable initial models. Since D_2 and D_3 come from the same source, we hypothesize that they share some similarities and aim to build on this assumption to benefit from transfer learning. For each of the datasets, we allocate 2/3 instances for training and the rest is equally divided into validation and calibration sets.

Exploratory Data Analysis and Pre-processing

In order to understand the problems and challenges we will face, we conducted a brief exploratory data analysis (see Appendix A.3). We had many web scraping errors in our data, that is why we decided to remove observations with wages less than 15000 and greater than 400000. In addition, we also removed duplicates and observations with missing values. Furthermore, we decided to lowercase all the text so that we lighten the vocabulary.

We concluded that the sequences are relatively long, and that the use of long memory RNNs was most likely a necessity. Furthermore, the sequences are short enough to fully fit into a BERT model, which is limited to 512-token long sequences. As far as salaries are concerned, there seems to be no problem except a slight skewness.

Model Architectures

The last layer of all our models consists of MLPs with 2 hidden layers. We use Leaky ReLu as non-linearity to avoid the vanishing gradient effect. For BLSTMs, we chose to keep the hidden state of the same dimension as the embedding vectors (300). Due to the bi-directionality, the MLP layers are twice as large as the hidden state, as shown in Figure 3. These values were chosen by a grid search on D_3. We decided not to use a conventional gradient descent algorithm to avoid its instability around local optima (Sutton, 1986; Ruder, 2016). Instead, we decided to use the Adaptive moment estimation (Adam), an adaptive learning rate algorithm introduced by Kingma and Ba (2014). Optimization is done on mini-batches of size 64.

The BERT model consists of 24 Transformer blocks with a hidden state of size 1024, 16 heads per block for a total of 336 million parameters. Similar to BLSTM, we add an MLP with 2 hidden layers on top of the NSP input. We chose a slightly different learning strategy than the previous one. We implemented a two-phase learning. The first phase consists in training only the MLP, then in a second phase we back-propagate through the whole network. Optimization is done on mini-batches of size 16. We have observed much better results by applying this method. A dropout of 10% is used in each feed-forward layer of the Transformer blocks.

In case of weight sharing, we experimentally found out that $\kappa = 1$, $\lambda = 3$ and $\mu = 3$ in (31) works well. For quantile regression, we decided to focus on 95% prediction intervals by setting $\alpha_l = 1 - \alpha_u = 2.5\%$.

Experimental Plan

First, we want to implement our models for the D_1 dataset to study their efficiency without being limited by small amount of data. Furthermore, we will investigate the consequences of reducing the amount of data in the models to determine whether this is consistent with the model developed for the D_3 dataset later on. For this purpose, we will use 15000, 5000 and 1000 data points to develop BLSTM and BERT models. We will analyze the prediction quality and the average width of the prediction intervals. In

a second section, we will tackle the modeling for the D_3 dataset, investigating double transfer learning and weight sharing.

4 Results

4.1 Models on the U.S. Dataset D_1

The amount of data required for machine learning depends on many factors, such as the unknown underlying function linking features and response variables, or the complexity of the learning algorithm. Sophisticated machine learning algorithms can learn highly non-linear relationship between features and response variables. Therefore, they also have a high variance, which means that the predictions will vary based on the data used to train them. The price for this flexibility is that they require a lot of training data. Since natural languages are so diverse and complex, it is not surprising that NLP tasks require large amounts of training data. To illustrate our point, we will study the effect of increasing the size of the training set in our models. In this section, we used BLSTM models with Word2Vec and GloVe, as well as BERT models. Our results before and after conformalization are detailed in Table 1 and Table 2.

As expected, the number of observations plays a major role on the quality of the models, as shown in Figure 12, 13 and 14. The best performing model in terms of RMSE is the BERT model, while the GloVe + BLSTM model produces the shortest intervals on average, with a satisfactory coverage. Furthermore, it seems clear that conformalization works accurately when the calibration set is large enough, as shown by the Coverage against sample size in Table 1 and 2. This is because we have a much better estimate of the error distribution defined in equation (13). It is not surprising that BERT performs well given the amount of data used to pre-train it. However, the results of the GloVe + BLSTM model for the prediction intervals are satisfactory and suggest that double transfer learning can provide benefits.

	RMSE	Interval width	Coverage
n=1000 / Constant model	**53847**	**201655**	**95,45**
n=5000 / Constant model	**56188**	**218474**	**94,54**
n=15k / Constant model	**56945**	**214728**	**94,77**
n=1000 / GloVe + BLSTM	**26868**	**172338**	**92,6**
n=5000 / GloVe + BLSTM	**18489**	**68310**	**92,8**
n=15k / GloVe + BLSTM	**11373**	**36322**	**92,5**
n=1000 / W2V + BLSTM	**28956**	**168811**	**90,3**
n=5000 / W2V + BLSTM	**20362**	**60949**	**91,9**
n=15k / W2V + BLSTM	**11174**	**46602**	**94,6**
n=1000 / BERT	**17384**	**66401**	**88,1**
n=5000 / BERT	**12711**	**45705**	**86,6**
n=15k / BERT	**9117**	**44972**	**93,1**

Table 1: Model performance for the D_1 validation set before conformalization, where Root Mean Squared Error (RMSE) and Interval width are expressed in dollars, and coverage in percentages. The RMSE is computed as in equation (6), and the interval widths are averaged over the entire validation set.

	RMSE	Interval width	Coverage
n=1000 / Constant model	**53847**	**160983**	**90,9**
n=5000 / Constant model	**56188**	**231959**	**93,97**
n=15k / Constant model	**56945**	**212013**	**95,06**
n=1000 / GloVe + BLSTM	**26868**	**245718**	**96**
n=5000 / GloVe + BLSTM	**18489**	**79443**	**96**
n=15k / GloVe + BLSTM	**11373**	**40029**	**94,6**
n=1000 / W2V + BLSTM	**28956**	**247386**	**98,3**
n=5000 / W2V + BLSTM	**20362**	**74444**	**96,39**
n=15k / W2V + BLSTM	**11174**	**45263**	**94,89**
n=1000 / BERT	**17384**	**111215**	**98,9**
n=5000 / BERT	**12711**	**59853**	**96,39**
n=15k / BERT	**9117**	**47901**	**95,6**

Table 2: Model performance for the D_1 validation set after conformalization, where Root Mean Squared Error (RMSE) and Interval width are expressed in dollars, and coverage in percentages. The RMSE is computed as in equation (6), and intervals are first conformalized, and then their average width is calculated on the whole validation set.

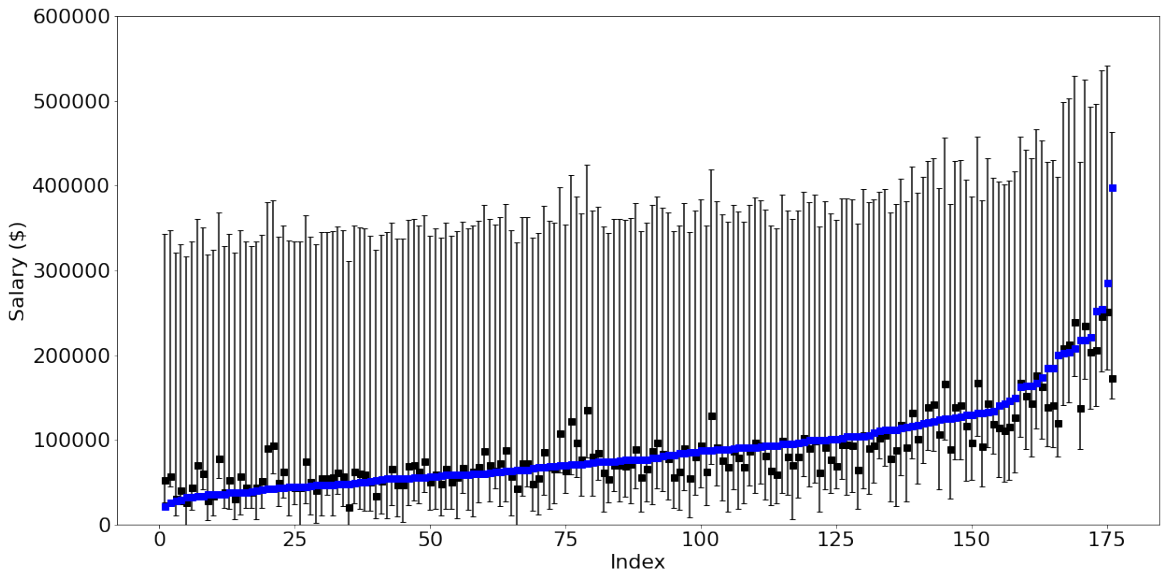


Figure 12: Predictions with conformalized prediction intervals on the validation set for the GloVe + BLSTM model, with $n = 1000$, where the predictions are in black and the true labels in blue. The salary is expressed in dollars and the index represents jobs with the lowest to highest salary.

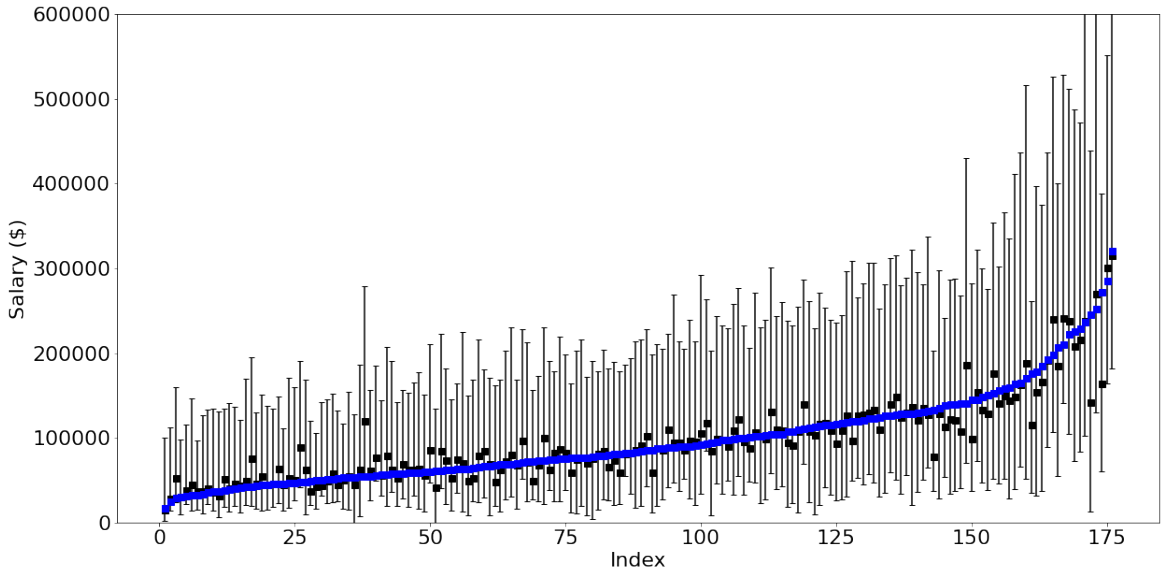


Figure 13: Predictions with conformalized prediction intervals on a subset of the validation set for the GloVe + BLSTM model, with $n = 5000$, where the predictions are in black and the true labels in blue. The salary is expressed in dollars and the index represents jobs with the lowest to highest salary.

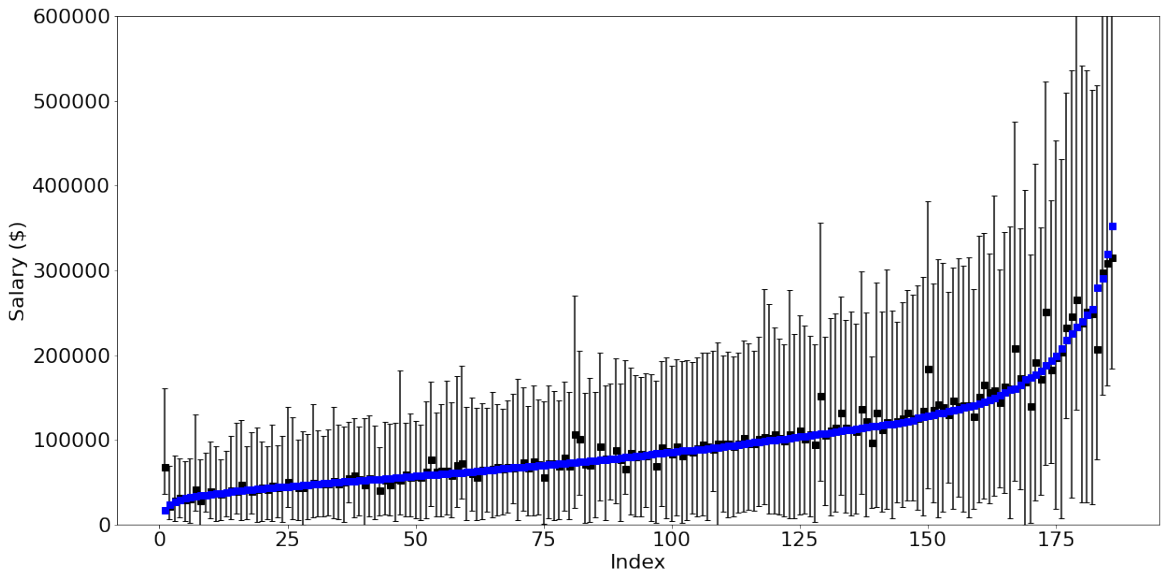


Figure 14: Predictions with conformalized prediction intervals on a subset of the validation set for the GloVe + BLSTM model, with $n = 15000$, where the predictions are in black and the true labels in blue. The salary is expressed in dollars and the index represents jobs with the lowest to highest salary.

We observe an inaccurate estimate of the upper bound of the prediction intervals when the sample size is small. We hypothesize that this is due to the asymmetry of the salary distribution. However, it would not be appropriate to simply remove the extreme observations from our data because they represent the reality of the labor market. Secondly, we observe that as the sample size increases, not only does this error decrease, but also that the prediction intervals are roughly proportional to the salaries. We believe this is because, in general, high-paying jobs also have high variability in pay.

It is difficult to visually distinguish the difference before and after conformalization, as the changes are generally small. The bottom line is that it is very effective when you consider the benefits in terms of coverage, as shown in Table 1 and 2. Figure 15 and 16 show the prediction intervals before and after conformalization of our BERT model, which performs best based on RMSE.

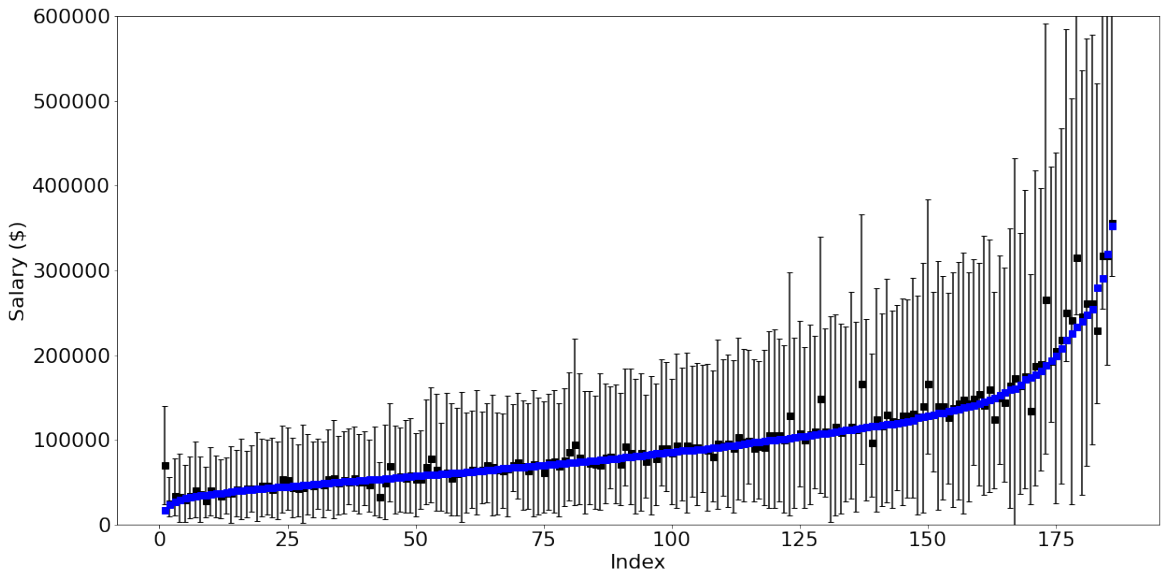


Figure 15: Predictions on a subset of the validation set for the BERT model, with $n = 15000$, where the predictions are in black and the true labels in blue. The salary is expressed in dollars and the index represents jobs with the lowest to highest salary.

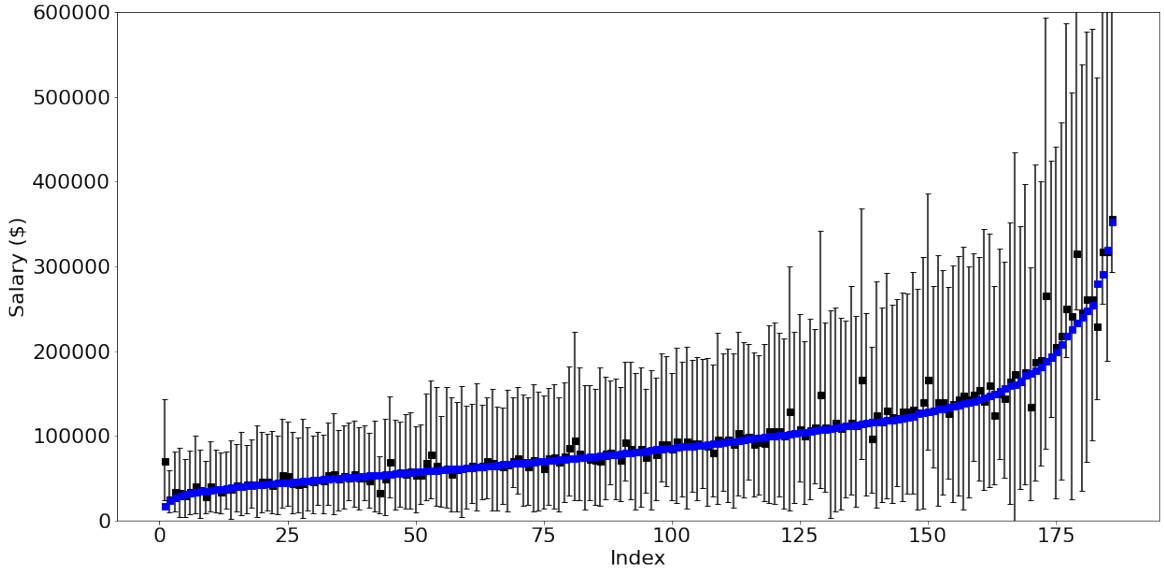


Figure 16: Predictions with conformalized prediction intervals on a subset of the validation set for the BERT model, with $n = 15000$, where the predictions are in black and the true labels in blue. The salary is expressed in dollars and the index represents jobs with the lowest to highest salary.

4.2 Models on the Swiss Dataset D_3

Now that we know what to expect from our model based on the training set size, we can move on to our main task. In this section, we show the results for all models considered in this analysis, including models based on weight sharing and double transfer learning. The first interesting phenomenon we observe is the limited contribution of Double Transfer Learning (DTL) in terms of RMSE minimization, as shown in Table 3 and 4. RMSE is only a summary statistic, and we should analyze the predictions against the true labels more carefully. Figures 17 and 18 show the predictions against the true labels with and without DTL.

We observe that the DTL provided an improvement for the prediction of extreme values. On the other hand, we lose precision around the values in the center of the distribution. We therefore conclude that DTL provided a better estimate for unusual salaries. In addition, the DTL also allows for greater variability of the prediction intervals while maintaining satisfactory coverage. We believe that these improvements are due to the fact that the model was previously trained on a wider variety of descriptions. The second interesting point is the improvement due to weight sharing. Indeed, we obtain the shortest prediction intervals with the GloVe + BLSTM model with weight sharing and DTL.

In contrast, weight sharing does not appear to have a positive effect on the BERT model. However, the results are satisfactory considering that we went from 1008 million parameters to 336 million. In addition, the BERT model gives us the lowest RMSE. Figure 19 shows the predicted against the true labels for this BERT model. We observe that the predictions of extreme values are also improved. In addition, the BERT model seems to maintain a better precision in the center of the distribution than the BLSTM model. However, the variability of the prediction intervals is lower compared to the BLSTM.

The last interesting point is the ability of conformalization to mitigate over-fitting. Although conformalization is much more effective when the calibration set is large, we still observe a strong improvement in coverage on complex models such as BERT, as shown in Table 3 and 4.

	RMSE	Interval width	Coverage
Constant model	**38671**	**158751**	**96**
W2V + WS + BLSTM	**30015**	**114016**	**96**
GloVe + WS + BLSTM	**28433**	**108156**	**94,89**
W2V + BLSTM	**28521**	**138257**	**92**
GloVe + BLSTM	**27315**	**124277**	**90,9**
W2V + WS + BLSTM + DTL	**27395**	**100122**	**95,39**
GloVe + WS + BLSTM + DTL	**28880**	**102586**	**94,3**
W2V + BLSTM + DTL	**31510**	**110341**	**96,6**
GloVe + BLSTM + DTL	**27193**	**103537**	**93,7**
WS + BERT	**21439**	**143427**	**99,4**
BERT	**21227**	**88111**	**90,9**

Table 3: Model performance for the D_3 validation set before conformalization, where Root Mean Squared Error (RMSE) and Interval width are expressed in Swiss francs, and coverage in percentages. The RMSE is computed as in equation (6), and the interval widths are averaged over the entire validation set.

	RMSE	Interval width	Coverage
Constant model	**38671**	**173276**	**97,71**
W2V + WS + BLSTM	**30015**	**130201**	**97,1**
GloVe + WS + BLSTM	**28433**	**129075**	**97,7**
W2V + BLSTM	**28521**	**137934**	**93,7**
GloVe + BLSTM	**27315**	**144081**	**95,39**
W2V + WS + BLSTM + DTL	**27395**	**120405**	**98,3**
GloVe + WS + BLSTM + DTL	**28880**	**112283**	**95,39**
W2V + BLSTM + DTL	**31510**	**123945**	**98,3**
GloVe + BLSTM + DTL	**27193**	**131031**	**98,3**
WS + BERT	**21439**	**118526**	**96,6**
BERT	**21227**	**114027**	**97,7**

Table 4: Model performance for the D_3 validation set after conformalization, where Root Mean Squared Error (RMSE) and Interval width are expressed in Swiss francs, and coverage in percentages. The RMSE is computed as in equation (6), and intervals are first conformalized, and then their average width is calculated on the whole validation set.

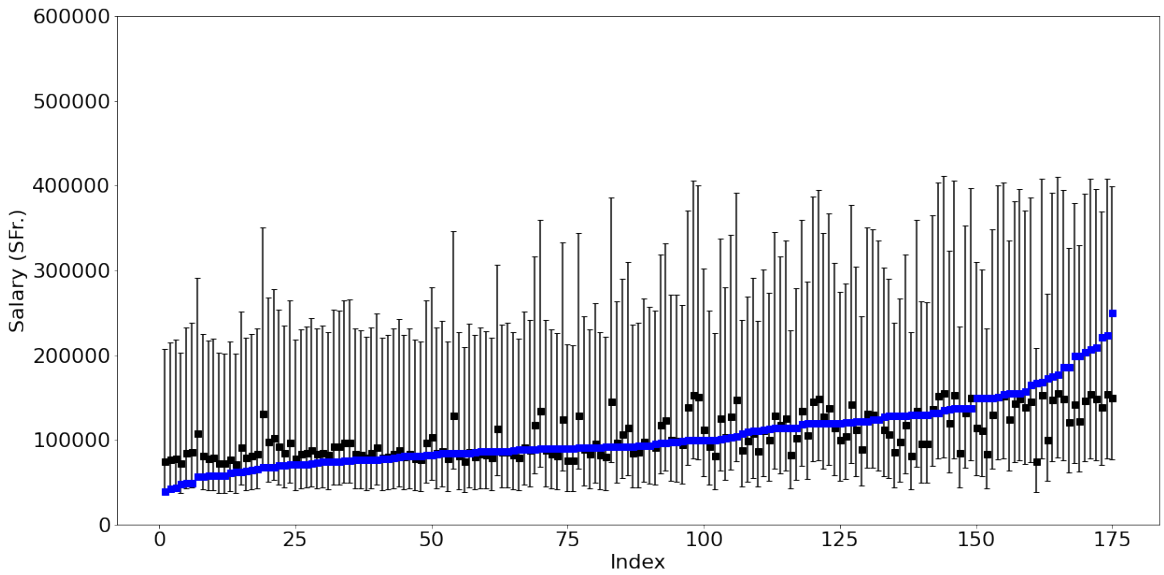


Figure 17: Predictions with conformalized prediction intervals on the validation set for the GloVe + WS + BLSTM model, where the predictions are in black and the true labels in blue. The salary is expressed in Swiss francs and the index represents jobs with the lowest to highest salary.

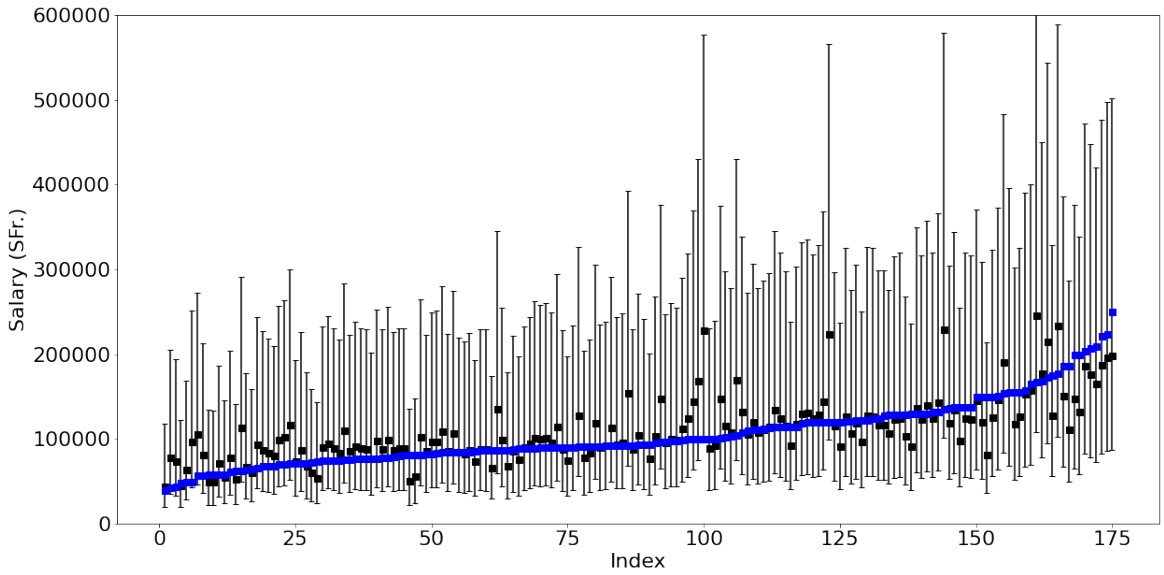


Figure 18: Predictions with conformalized prediction intervals on the validation set for the GloVe + WS + BLSTM + DTL model, where the predictions are in black and the true labels in blue. The salary is expressed in Swiss francs and the index represents jobs with the lowest to highest salary.

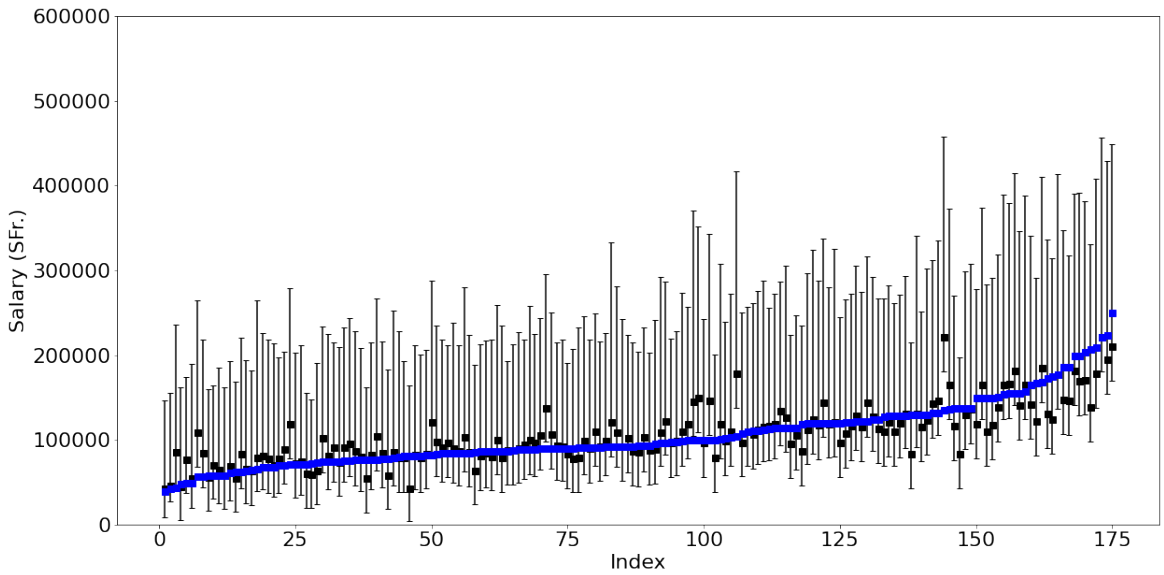


Figure 19: Predictions with conformalized prediction intervals on the validation set for the BERT model, where the predictions are in black and the true labels in blue. The salary is expressed in Swiss francs and the index represents jobs with the lowest to highest salary.

5 Discussion and Conclusion

The objective of this master thesis was to use natural language processing models to predict the expected salary of an individual from the description of the job he or she is applying for, in the Swiss labor market. In addition, we were also interested in prediction intervals. To do so, we proposed to use pre-trained BLSTM and BERT models to produce both expected wage predictions and prediction intervals using conformalized quantile regression. Due to the limited amount of data available for the Swiss labor market, we decided to use U.S. labor market data to pre-train our BLSTM models. As for BERTs, we used models that were already pre-trained on a large amount of unlabeled data. Given that natural language processing models tend to be numerically expensive and that our different tasks likely shared common explanatory variables, we also explored the idea of weight sharing. To this end, we developed models to minimize a linear combination of pinball and RMSE loss functions. In a first step, we implemented our models on a large U.S. labor market dataset, in order to study their effectiveness without being limited by a small amount of data. Then, in a second step, we applied our transfer learning and weight sharing methods to produce our models for the Swiss labor market.

The results of our models for the U.S. labor market showed that we can obtain satisfactory results if we were given a large amount of data. As for the models for the Swiss data, we found that double transfer learning improved the prediction of extreme values and allowed us to have a greater variability in the bounds of our prediction intervals while maintaining a satisfactory coverage. The other interesting point we noticed was the improvement due to weight sharing. Indeed, we produced the shortest prediction intervals on average with our BLSTM model with double transfer learning and weight sharing. For our BERT models, we obtained the lowest RMSE without weight sharing. However, we showed that we can obtain similar results with weight sharing, and thus reduce the size of our models by 672 million parameters. As for conformalization, we have also seen that in addition to generating prediction intervals with guaranteed theoretical coverage, it also allows us to reduce the impact of over-fitting and produce better coverage precision at the desired level.

While we can improve results with weight sharing and transfer learning, the size of the training data appears to be a key factor in achieving better results. Moreover, a larger number of observations would allow us to increase the size of the calibration set and thus obtain more accurate validation coverages. Besides the issue of data quantity, we also believe that our models could be more efficient if we had several observations for each job. For example, we believe our models could better identify the key factors that determine the difference in pay between two similar jobs. We were unable to obtain this type of data because no job advertising company provides downloadable, up-to-date data of this quality. In fact, this data is freely consultable but difficult to scrape because of CAPTCHA security measures implemented by job advertising websites.

Regarding the models presented in this thesis, one could suggest some ideas for possible improvements. For example, one could modify the architecture of our BERT model. Classically, one uses only the input of the NSP task to produce predictions. For example, the input of the Mask ML task could be exploited to increase performance by using a CNN. For BLSTMs, one could also try to use Gated Recurrent Units (GRUs) (Chung et al., 2014), a type of recurrent cell similar to LSTMs with fewer parameters.

A Appendix

A.1 Pinball loss

Proof. Finding the optimal function $q_{(\theta,\mathbf{w})}(X_i)$ according to $\mathcal{P}(\theta,\mathbf{w})$ lead to a θ-quantile regression.

$$\begin{aligned}\min_{q_{(\theta,\mathbf{w})}(X_i)} E[\mathcal{P}(\theta,\mathbf{w})] &= \min_{q_{(\theta,\mathbf{w})}(X_i)} \int_{-\infty}^{\infty} \mathcal{P}(\theta,\mathbf{w}) f(Y) dY \\ &= \min_{q_{(\theta,\mathbf{w})}(X_i)} (\theta-1) \int_{-\infty}^{q_{(\theta,\mathbf{w})}(X_i)} \left(Y - q_{(\theta,\mathbf{w})}(X_i)\right) f(Y) dY \\ &\qquad + \theta \int_{q_{(\theta,\mathbf{w})}(X_i)}^{\infty} \left(Y - q_{(\theta,\mathbf{w})}(X_i)\right) f(Y) dY. \end{aligned} \tag{32}$$

Taking the derivative w.r.t. $q_{(\theta,\mathbf{w})}(X_i)$ we have

$$\frac{\partial E[\mathcal{P}(\theta,\mathbf{w})]}{\partial q_{(\theta,\mathbf{w})}(X_i)} = (1-\theta) \int_{-\infty}^{q_{(\theta,\mathbf{w})}(X_i)} f(Y) dY - \theta \int_{q_{(\theta,\mathbf{w})}(X_i)}^{\infty} f(Y) dY. \tag{33}$$

By setting the derivative to zero and simplifying, we obtain

$$F\left(q_{(\theta,\mathbf{w})}(X_i)^{\star}\right) = \theta, \tag{34}$$

where F is the cumulative density function of Y.

■

A.2 Example of job description

To better understand the data, here follows three descriptions drawn from D_1, D_2 and D_3, respectively.

"The Entry Structural Engineer performs analysis of building materials for use in construction. Designs load-bearing structures or structural elements, such as buildings, bridges, or roadways. Being an Entry Structural Engineer determines cause of structural failures, damages, and defects through site investigations. Develops blueprints or specifications for use during construction and ensures all projects comply with applicable codes and regulations. In addition, Entry Structural Engineer provides reports detailing investigations and assessment of damages to the structure. Requires a bachelor's degree. Typically reports to a supervisor or manager. Working as an Entry Structural Engineer typically requires 0-2 years of related experience. Works on projects/matters of limited complexity in a support role. Work is closely managed."

"Accountants perform financial calculations for companies in a wide variety of fields. Some common duties include creating sales and cash flow reports, administering payroll, keeping balance sheets, carrying out billing activities, managing budgets and keeping inventory. The accountant may also be responsible for filing taxes for the company, as well as reviewing past reports to generate income forecasts. Occasionally, internal audits must be carried out to make sure that the various areas of the company are performing as expected; the accountant must also make sure that staff members are adhering to company policies and relevant laws. The accountant should be able to create accurate, detailed reports to illustrate data; sometimes, these reports have to be presented to management. The accountant may oversee the financial transactions of one department or multiple departments within their organization. A bachelor's degree in accounting is required for this position, as is status as a certified public accountant (CPA). Previous accounting experience is generally required or preferred as well. Knowledge of accounting software such as Quickbooks and Microsoft Excel is needed. Additionally, many of the accountant's tasks are performed independently so it is essential to be self-motivated; however, collaboration is necessary, and the accountant must be able to work as part of a team."

"At larger companies and corporations, the director of compensation and benefits is a high-level executive position in human resources. This person is tasked with overseeing the fair wage structure offered to employees, all benefits and insurance, and the accurate compensation for any work-related injuries or accidents. The director is expected to set a wage scale that is competitive but also company-friendly, while assembling a benefits package that will help bring good, talented workers to the company. The director of compensation and benefits will typically be a person with extensive experience in human resources and company finance. The director will normally work with a staff to identify and classify every strata of jobs in the company. This person will then work with human resource metrics to help determine scarcity and surplus of qualified workers in those positions. The director then works with the financial and budgeting sectors of the company to determine the parameters of compensation to be offered for the various job duties within the organization. As an executive, this person will typically oversee

a staff involved in collecting data from within the company and from the industry at large. As with most high- level executives, most employers will look for persons with post-graduate degrees in business management for this position, and many companies may prefer to promote from within. A director of compensation and benefits typically works regular business hours during the week in an executive office."

A.3 Exploratory Data Analysis

	Descriptions length			Responses		
	D_1	D_2	D_3	D_1	D_2	D_3
Count	14464	6331	1050	14464	6331	1050
Mean	141	258	255	94490	66646	104275
Std	30	40	41	54540	35316	40108
Min	23	103	103	17215	16800	20000
Max	295	505	468	398100	395295	400000

Table 5: Summary Statistics.

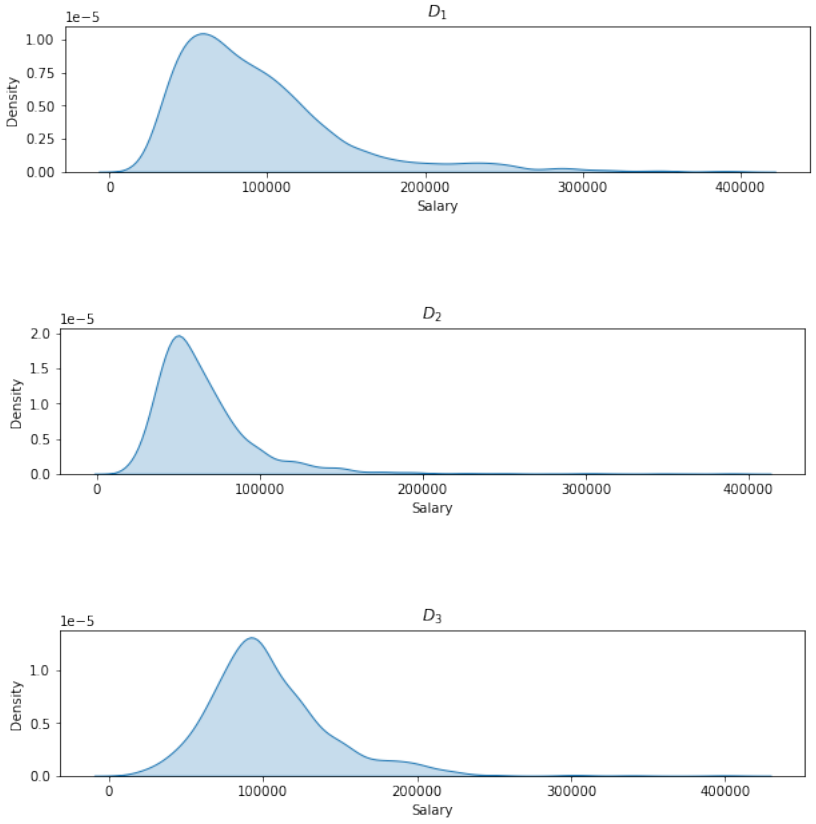


Figure 20: Salary distribution for D_1, D_2 and D_3.

References

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks, 157-166*, 1994.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems, 303-314*, 1989.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

J. Giordano and J. Gaffney. Salary Comparison, Salary Survey, Search Wages. https://www.payscale.com/, 2002. Accessed: 28 April, 2021.

Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722*, 2014.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation, 1735-1780*, 1997.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning, 448-456*. PMLR, 2015.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv:2105.03824*, 2021.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks, 861-867*, 1993.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning, 1310-1318*. PMLR, 2013.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 1532–1543*, 2014.

K. Plunkett. Unlock the Power of Pay. https://www.salary.com/, 1999. Accessed: 10 April, 2021.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*, 2016.

P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv:1806.03822*, 2018.

Y. Romano, E. Patterson, and E. J. Candès. Conformalized quantile regression. *arXiv:1905.03222*, 2019.

X. Rong. word2vec parameter learning explained. *arXiv:1411.2738*, 2014.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

M. Sahlgren. The distributional hypothesis. *Italian Journal of Disability Studies, 33-53*, 2008.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 1929-1958*, 2014.

R. Sutton. Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 823-832*, 1986.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.

R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation, 490-501*, 1990.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision, 19-27*, 2015.