Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Computer Science
21 | LIU-IDA/LITH-EX-A--21/040--SE

Extracting Salient Named Entities from Financial News Articles

Extrahering av centrala entiteter från finansiella nyhetsartiklar

David Grönberg

Supervisor : Ehsan Doostmohammadi
Examiner : Marco Kuhlmann

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

This thesis explores approaches for extracting company mentions from financial news articles that carry a central role in the news. The thesis introduces the task of salient named entity extraction (SNEE): extract all salient named entity mentions in a text document. Moreover, a neural sequence labeling approach is explored to address the SNEE task in an end-to-end fashion, both using a single-task and a multi-task learning setup. In order to train the models, a new procedure for automatically creating SNEE annotations for an existing news article corpus is explored. The neural sequence labeling approaches are compared against a two-stage approach utilizing NLP parsers, a knowledge base and a salience classifier. Textual features inspired from related work in salient entity detection are evaluated to determine what combination of features result in the highest performance on the SNEE task when used by a salience classifier. The experiments show that the difference in performance between the two-stage approach and the best performing sequence labeling approach is marginal, demonstrating the potential of the end-to-end sequence labeling approach on the SNEE task.

Acknowledgments

First and foremost, I would like to thank Andra AP-fonden for the thesis opportunity. In particular, I would like to thank Jan Lennartsson and Johan Wiebe, not only for their great assistance and ideas, but also for the inspiring conversations regarding the potential of NLP in finance. In addition, I would like to thank my supervisor Ehsan Doostmohammadi, as well as my examiner Marco Kuhlmann for their constructive feedback and useful suggestions throughout the thesis work.

Contents

List of Figures

List of Tables

Abbreviations

BiLSTM Bidirectional Long Short Term Memory Networks.

CRF Conditional Random Field.

IOB Inside-Outside-Beginning.

LSTM Long Short Term Memory Networks.

MTL Multi-task Learning.

NEL Named Entity Linking.

NER Named Entity Recognition.

NLP Natural Language Processing.

REL Radboud Entity Linker.

RNN Recurrent Neural Network.

SNE Salient Named Entity.

SNEE Salient Named Entity Extraction.

1 Introduction

The following chapter presents the aim and motivation of the thesis, as well as the formulated research questions and delimitations.

1.1 Motivation

Due to the exponential growth of generated information online, reading and filtering texts of large collections turns into a challenging task. Meta-data can be used effectively to assist in filtering irrelevant documents. However, a large majority of documents does not have meta-data attached to it, thus there is a need to accurately generate this meta-data automatically. Moreover, textual information from the media domain, such as news stories, blog posts, and other unstructured text data, can have an observable effect on the financial markets [1]. The vast amount of information produced each day places pressure on investors to continually stay updated with the latest news and act on the underlying signals. Furthermore, information from these media outlets can also help investigate companies and examine their activity, such as what ventures a company is involved in and with whom.

Andra AP-fonden (AP2) is one of the five buffer funds within the Swedish pension system. The fund has a well diversified strategic portfolio consisting of investments in a diverse set of asset classes such as global equities, corporate credit bonds, sovereign debt and private equity. The National Pension Insurances Funds Act regulates AP2's operation, meaning that AP2 is required to only conduct responsible and sustainable investments. As a result, AP2 must judge whether or not a company is involved in violating any particular convention and if AP2 should invest in the company. This creates a desire for AP2 to stay updated and informed about the actions and activity of the companies in their portfolio. One way of aiding the gathering of information about companies is to follow financial news. However, because of the large number of news articles generated every day, it is challenging and tedious to filter through the noise and extract relevant news articles worth closer attention.

Natural language processing (NLP) has recently attracted significant attention due to its ability to automate knowledge extraction from text. Named Entity Recognition (NER) is a task in NLP with the goal of recognizing and categorizing named entities in unstructured text data into pre-defined categories, such as locations, people, and organizations [2]. NER can be used by AP2 to identify articles that mention specific companies of interest by performing NER on the article body text and investigating if any of the recognized named entities match

a company of interest. Moreover, a NER model can be trained to only find named entities within the 'Organizations' category to narrow down the task. After the company mentions have been extracted from a document, Named Entity Linking (NEL) can be used to link the company mentions to the corresponding company's unique identifier in a knowledge base and map that unique identifier to the news article.

At first glance, this approach can be sufficient for the task at hand. However, financial news articles often mention several companies in a story, giving rise to multiple recognized companies, where not all play a prominent role in the news presented in the article. Consider the following example:

> 'Health insurer **Company 1** on Thursday posted higher quarterly profit as it said medical cost trends moderated and enrollment grew. Net income at the Hartford, Connecticut-based company rose to $476 million, or 85 cents per share, from $373 million, or 62 cents per share, a year earlier. **Company 1** shares have fallen about 18 percent this year, more than most health-insurance peers, as its quarterly results earlier this year disappointed investors. The stock changes hands at about 12.5 times next year's earnings estimates, less than for rivals **Company 2** and **Company 3**.'

Figure 1.1: News article example with the headline: *Company 1 quarterly profit rises*.

The news article in figure 1.1 illustrates an example where multiple companies are mentioned in the body text. The article's headline and content suggest that *Company 1* is the article's primary focus. *Company 2* and *Company 3* are briefly mentioned in the last sentence. Extracting *all* the company mentions in a news article will lead to unnecessary and incorrect linking. Therefore, it would be beneficial to only recognize the *salient* companies in news stories, i.e. companies that play a prominent role in the news, and disregard companies that are not important to the news. This task is closely related to extracting salient entities in text documents, referred to as *salient entity discovery* or *document aboutness* [3], where entities are a superset of named entities.

Previous approaches to salient entity discovery often consist of a two-stage approach, by first forming a list of candidate entities and then pruning the list using supervised techniques that often uses hand-crafted features and external knowledge bases to derive the salient entities. However, using hand-crafted features might fail to exploit the document's full underlying semantics. Moreover, using the two-stage approach requires optimizing several stages (the candidate generation stage and the classification stage). Therefore, language models capable of catching deep semantics in text can perhaps achieve a higher performance in extracting the salient named entities from a text document. Furthermore, approaching the task of extracting salient named entities as a sequence labeling task would make it possible to model the task as an end-to-end task, removing the requirement to optimize several stages.

1.2 Aim

This thesis aims to study different approaches for the task of extracting salient named entities from financial news articles, henceforth referred to as Salient Named Entity Extraction (SNEE). In this context, salient named entities refer to companies and organizations mentioned in the news article, which are the news article's main focus.

Three different approaches are studied, where each approach relates to previous work done in similar research areas. More specifically, the following approaches are studied:

- **Approach 1**: A two-stage approach using a NER and NEL model to generate candidate named entities and a binary classifier utilizing hand-crafted features to prune the candidate list.

- **Approach 2**: A single-task neural sequence labeling approach that treats the task of SNEE as a sequence labeling task.
- **Approach 3**: A multi-task learning approach using neural sequence labeling.

The two-stage approach is inspired by previous work in salient entity discovery, while the sequence labeling approaches are based on previous work in keyword extraction and other sequence labeling tasks.

The purpose of this thesis is to study how the task of extracting salient named entities from news articles can be approached, and if the task can be solved in an end-to-end fashion using sequence labeling, effectively eliminating the need for an external knowledge base and hand-crafted features. Furthermore, this study explores how a dataset can be automatically constructed for the task of SNEE in news articles and creates a gold-standard corpus.

1.3 Research questions

The following research questions and their corresponding answers will form the basis of this thesis:

1. *Which combination of hand-crafted textual features yield the highest performance when used by a classifier for classifying salient named entities?*
2. *How does the sequence labeling approach perform on the salient named entity extraction task compared to the two-stage approach?*
3. *How does the performance of the sequence labeling models change when a multi-task learning approach is taken and an auxiliary named entity recognition task is added to the model?*

The first research question aims to study what combination of hand-crafted features benefit a classifier in the classical two-stage approach in the task of SNEE. Features proposed in related work on salient entity discovery are considered for evaluation. The second research question aims to study if the salient named entity extraction task can be modelled as an end-to-end sequence labeling task and if it can outperform the classical two-stage approach. The third research question aims to investigate if the end-to-end approach can be further improved by introducing an auxiliary task in a multi-task learning setup.

1.4 Delimitations

In this thesis work, the task of salient named entity extraction will be restricted to named entities of the Organization type, i.e., companies, political groups, government bodies, and public organizations. The evaluation of the studied approaches will be restricted to the domain of financial news articles. Furthermore, the evaluation of the approaches will focus on their performance in terms of precision/recall/f1-score. Thus, the training time, inference time and complexity of the approaches will not be studied.

Because there do not exist any existing datasets for the SNEE task, this study relies on the assumption that the salient companies in a news article are mentioned in the article's headline when curating a training set for the proposed approaches to train on.

2 Theory

The following chapter introduces the concepts and theory needed to realise the study. Moreover, previous work in similar research areas is introduced, together with a description of existing datasets for related tasks.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a sub-field in artificial intelligence that refers to the task of automatically manipulating natural language, by combining practices from the computer science, mathematics and linguistics domain. There are many tasks that can be approached using NLP. Some successful examples include language translation, sentiment classification of text and Named Entity Recognition [4, 5, 6].

2.1.1 Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying expressions in structured and unstructured documents that refer to named entities, such as people, geographical locations, organizations, and companies; and categorizing the mention into pre-defined categories [2]. These categories depict the type of the named entity, for example, if the mention is a person or a company. Consider the following sentence:

> *Google CEO Sundar Pichai buys Manhattan penthouse.*

The sentence contains in total three named entity mentions. *Google* is an organization (ORG), *Sundar Pichai* is a person (PER), and *Manhattan* is a geographical location (LOC). NER is often a necessary precursor for many information extraction tasks and NLP applications, such as question-answering, knowledge-base population, and text summarization. Furthermore, NER is not only restricted to the general domain. For example, in biomedicine, the entities of interest are genes and gene products, and in the homeopathic domain, drug names and diseases are recognized as entities [7].

Earlier NER systems relied heavily on handcrafted rules, which provided reasonable results but were often domain-dependent. Modern systems often approach the task of NER as a sequence labeling task and rely on supervised machine learning algorithms which overcome

the drawbacks of handcrafted systems. While recent neural NER models show satisfying results, there are still challenging cases that these models struggle with, such as nested entities and ambiguity in the text.

2.1.2 Entity Linking

Entity Linking (EL), or Named Entity Linking (NEL), is the task of identifying mentions of named entities and assigning unique identifiers to said entities using an underlying knowledge base [8]. EL is often divided into two tasks, NER and Entity Disambiguation (ED), where NER identifies the named entities and ED maps the found named entities to unique identifiers in a knowledge base. Consider the example sentence in the previous section, a NER-tagger would have identified the first word in the sentence to be a named entity of type Organization. However, it does not tell us *which* organization it refers to. This is the task of ED. A knowledge base (e.g., Wikipedia) is searched using different heuristics to retrieve the unique identifier. In the case of using Wikipedia as a knowledge base, a unique identifier can be the corresponding Wikipedia page of a named entity. There are several approaches to NEL, where some systems use the semantics among other features derived from the text to disambiguate, while other use knowledge graphs and connections to find the unique identifier.

2.2 Sequence Labeling

Sequence labeling is the task of assigning a label to each value in the sequence. A typical example of a sequence labeling task is part-of-speech tagging, where the task is to assign a part-of-speech tag to each word in a sentence. Other examples include word segmentation, entity mention detection and named entity recognition.

Sometimes a word can be part of a chunk. For example, in named entity recognition, *European Union* consists of two words but refers to a single named entity. Thus, there is a need to tag each word in the chunk as a part of a label. The IOB-format (Inside-Outside-Beginning), introduced in 1995 [9], is a commonly used format for sequence labeling and is used to denote the beginning, inside and outside of a chunk. For example, in a NER task, the sentence *The European Union is a political union* would be labeled as follows:

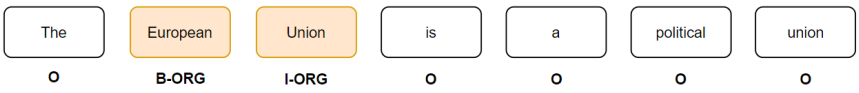

Figure 2.1: Sequence labeling example with NER labels for each token in the sequence.

In this example, *European* is labeled with the B-prefix, representing that it is the beginning of a named entity. The I-prefix represents that the word is part of the previous label (in this case ORG) and the O label indicates the the word is not part of a named entity.

2.2.1 Evaluation Metrics in Sequence Labeling

Sequence labeling is often formulated as a classification task, thus the commonly used evaluation metrics are the *precision*, *recall* and *f1-score*, which are introduced in chapter 2.3.5. However, there are many scenarios to consider when comparing the sequence labeling output to the ground truth. A model can predict the incorrect boundaries for the IOB-tagging format, or the incorrect category label. Therefore, the evaluation should be able to incorporate these scenarios. The International Workshop on Semantic Evaluation (SemEval) introduced four different ways to measure precision/recall/f1-score results with varying strictness:

- Strict: exact boundary surface string match and type.

- Exact: exact boundary match over the surface string, regardless of the type.
- Partial: partial boundary match over the surface string, regardless of the type.
- Type: some overlap between the system tagged entity and the gold annotation is required.

This evaluation schema was originally designed for Named Entity Recognition, but can be generalized to other sequence labeling tasks.

2.3 Classifiers

In machine learning, a classifier refers to a model that, given an input vector X predicts a value $\hat{y}$ from a pre-defined set of class labels. The task can be in the form of binary classification, e.g. where a model predicts if a customer will default on a bank loan or not, or the task can be a multi-class classification problem, for example, handwritten digit recognition. Classifiers fall into the category of supervised learning models, thus these models require training data containing the ground truth class labels for each training example, in order to learn the relation between the input and output values. A wide range of machine learning classifiers exists, where each classifier's performance depends on the nature of the classification task. The classifiers utilized in this thesis are *Random Forests* and *Gradient Boosted Decision Trees*, as presented below.

2.3.1 Random Forest

Random Forest is a machine learning algorithm that can be used for both classification and regression tasks. It is based on the simple *Decision tree*, a tree-like model where each node represents a test of a feature and each branch represents the outcome of the test. The resulting decision tree is learned by learning simple decision rules, recursively partitioning the feature space of the training set, deciding which conditions are best to split at by using different metrics such as e.g. *Gini index* and *Information gain* [10]. The Gini Index effectively measures how much noise a condition has, while Information Gain determines how much information about the target class is gained for each condition. See figure 2.2 for an example of a simple decision tree. As the figure shows, each node has a linear decision boundary with a binary outcome. The decision tree is a construction of many linear boundaries, making the decision tree non-linear.

Random Forest consists of a collection of decision trees, forming an *ensemble* to derive a prediction given an input. In classification tasks, each individual tree in the Random Forest makes a prediction, and the prediction with the majority vote becomes the model's final prediction [11]. For a Random Forest to be effective and perform well, the ensemble's individual trees should have low correlation between each other. If all the individual trees use the same features in the same way, the ensemble approach becomes redundant. One way Random Forest ensures uncorrelated decision trees is that each decision tree randomly samples from the training set, hence the name Random Forest. Another way Random Forest lowers correlation across trees is that each decision tree is constructed using a randomly sampled subset of features, which results in more diversification.

2.3.2 Gradient Boosted Decision Trees

Gradient boosting is a machine learning technique used to sequentially create a ensemble model based on multiple weak models [12]. In classification, the ensemble classifier is constructed by iteratively adding weak classifiers that are specifically fit to correct the prediction errors made by the previous classifiers. Data points missclassified by the previous classifiers are assigned a higher weight, while correctly classified data points loose weight. The added

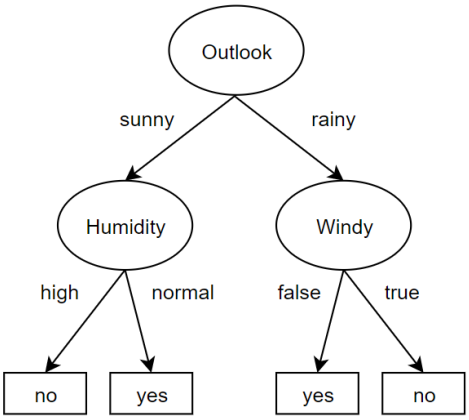

Figure 2.2: Example of a binary classification using a decision tree to decide if someone should play badminton outside depending on the weather.

weak classifiers will attend more to data points assigned a higher weight, i.e., each new weak classifier fits to the residuals from the previous step, improving the model's overall accuracy. The term *gradient* in gradient boosting stems from the way the ensemble model learns. In each step, after calculating the loss of the previous version of the ensemble, a classifier is added to the model to reduce the loss. This is done by functional gradient descent, where the added weak classifier is parameterized and modified, conditioned on a learning rate, to move the gradient in the right direction, reducing the residual loss.

2.3.3 Feature Selection

As previously mentioned in section 2.3, a classifier uses a feature vector X to predict an outcome $\hat{y}$ where X consists of n feature variables. Choosing these features can greatly impact the performance of the model and should be done with care. Feature selection is the process of choosing which feature variables should be in the feature vector X. The process is often intended to reduce the number of feature variables to a set of variables that are believed to be the most useful to a model, while losing as little information as possible in the process.

2.3.4 SHAP values

Understanding the inner workings of machine learning models is often of great importance. However, it becomes increasingly more difficult to retain a high explainability as the models become more complex, e.g. in ensemble models and deep learning. The need for explaining complex models led to the development of SHAP (SHapley Additive exPlanations), a framework for interpreting predictions from complex models [13]. SHAP uses Shapley values from coalition game theory. In the domain of machine learning, the features act as players in a coalition and the Shapley value identify how much each feature contributes to the predicted value. The Shapley value for a specific feature i, given a prediction p is calculated as follows:

$$\phi(p) = \sum_{S \subseteq N/i} \frac{|S|!(n - |S| - 1)!}{n!}(p(S \cup i) - p(S))$$

where S is the coalition and n is the total number of features. The above function essentially calculates the prediction of the model without the feature i, along with the the models

prediction with the feature i, to finally calculate their difference. The calculated difference is the effect the feature has on the prediction.

2.3.5 Evaluation of Classifiers

There are several methodologies taken when evaluating classifiers. The evaluation metrics relevant for one classification problem may not be meaningful in another problem. The most commonly used evaluation metrics are *accuracy*, as well as *precision*, *recall* and the *f1-score*. Precision is the proportion of correctly categorized samples out of all the positively predicted sample

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

where TP are the true positives classified by the classifier and FP are the false positives. True positives are samples classified as positive that are in fact positive, and false positives are samples classified as positive that actually are negative. For example, a true positive is when a model correctly predicts that an email is spam, and a false positive is when the model predicts that an email is spam, that is not. Thus, precision measures the percentage of predicted positives that are correctly classified.

Recall can be seen as a measure of how well a model finds data points of interest. Thus, for all emails that are spam, recall measures how many that are in fact identified to be spam by a classifier.

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

Recall measures the number of true positives divided by the number of true positives plus the number of false negatives. False negatives are samples classified as negative that are in fact positive. The f1-score is the harmonic mean of precision and recall.

$$f1\text{-}score = 2\frac{precision \cdot recall}{precision + recall} \tag{2.3}$$

The f1-score is a more sophisticated measure compared to a simple mean because it punishes extreme values. For example, if precision is 1 and recall is 0, the resulting f1-score would amount to 0, while the mean would result in 0.5. A high f1-score implies both a high precision and a high recall value. Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

Although accuracy is a good indicator of how good a classifier is, it rarely provides sufficient information for conducting a thorough evaluation of a classifier's performance.

2.4 Artificial Neural Networks

Artificial Neural Networks (ANN) are based on a collection of connected basic processing units referred to as neurons. ANN's are loosely inspired by the human brain, where hundreds of billions of connected neurons process information in parallel. ANN's consist of an input layer, output layer and a number of hidden layers, where each layer has a number of neurons. Each connection in the ANN is associated with a weight, which is learned in the training phase. The inputs from the incoming connections to a neuron are multiplied with their corresponding weight and, together with a bias term, fed through a continuous mathematical function referred to as an *activation function*. The output of the activation function is

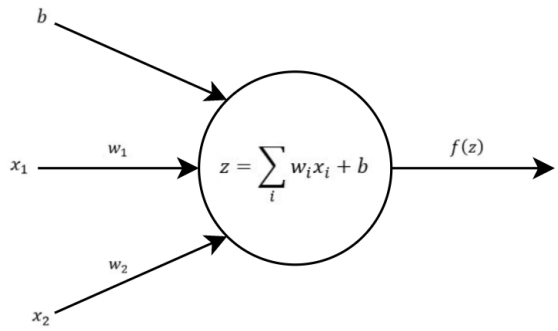

Figure 2.3: Example of a neuron in a Neural Network. x_1, x_2 are inputs, w_1, w_2 are the inputs corresponding weights, b is a bias term and $f(z)$ is an activation function.

then fed forward in the network. An example of a neural network neuron is shown in figure 2.3.

The purpose of the activation function is to introduce nonlinearity into the network and bound the neuron's value. A common nonlinear activation function is the *logistic function*, with the formula shown in equation 2.5. In the logistic function, also referred to as the sigmoid function, the input to the function is transformed into a value between 0 and 1. Inputs that are much larger than 1 are bounded to 1, while values much smaller than 0 are bounded to 0.

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

There exist several other activation functions, where each function is designed to serve a specific purpose. The simplest activation function is the linear activation function, where no transform is applied at all. A network comprised of only linear activation functions is simple to train, however, such networks are unable to learn complex mappings. Another activation function is the hyperbolic tangent function, which is similar to the logistic function and outputs values between -1 and 1. A known problem with the logistic and hyperbolic tangent function when training ANNs is that they are insensitive to large value changes of the input and are prone to saturate. Once saturated, it becomes difficult to update the weights in order to train the network. The Rectified Linear Activation Unit (ReLU) [14] is a activation function that counters the risk of saturation.

$$\phi(z) = max(0, z) \tag{2.6}$$

As shown in equation 2.6, ReLU outputs the input value if the input value is positive and outputs zero otherwise. ReLU has been heavily adopted in the area of deep learning, i.e. when dealing with networks that have multiple hidden layers.

2.4.1 Loss Function

The goal of training an ANN is to learn the weights of the network so that the output approximates the function $f(X)$ for the training inputs X. This is achieved by minimizing a *loss function*. The loss function can be defined in many ways and depends on the nature of the problem. It should be designed to serve as a differentiable heuristic for the metric that is to be optimized. The most commonly used loss function in multi-class classification tasks is the

categorical cross-entropy loss. The categorical cross-entropy loss of one prediction example is given by equation 2.7.

$$Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i \tag{2.7}$$

where $\hat{y}_i$ is the i$^{\text{th}}$ prediction from the model, y_i is the i$^{\text{th}}$ target value and n is the number of classes. The training of the network consists of updating the weights to minimize the loss function. The process of iteratively updating the weights to minimize the loss is accomplished using backpropagation.

2.4.2 Backpropagation

The backpropagation algorithm calculates the partial derivative $\delta_{\mathcal{L}} / \delta_w$ of the loss function $\mathcal{L}$ with respect to any weight w or bias in the network. The gradients indicate how quickly the loss changes when the weights are adjusted. This is done through out the whole network, layer by layer, starting with the layer closest to the output and backpropagating the error backwards through the network, slightly modifying the network's weights. The process is repeated until the loss no longer decreases or reaches a desirable value.

2.4.3 Optimizer

The repeated updating of the weights in neural networks is done by optimizers. There exist several different optimizers, where all use different methods in how they update the weights. *Gradient Descent* is one of the simpler optimizers that update the weights after each *epoch*, i.e. after each full iteration of the whole training dataset. After an epoch, the loss is calculated and backpropagation is done to update the weights. The *learning rate* controls how much the weights should change in each update.

The Adam optimizer [15] is an adaptive learning rate optimizer that uses individual learning rates for each weight in the neural network, as apposed to Gradient Descent which uses a single learning rate for all weights. Adam uses estimations of first (mean) and second (uncentered variance) moments of the gradient to adapt the learning rate of each weight. The moments of the gradient are calculated by storing an exponentially decaying average of past squared gradients. The hyperparameters β_1 and β_2 control the exponential decay rate of the moving averages.

2.4.4 Recurrent Neural Networks

Traditional neural networks have no memory of past input and only consider the current input when making a prediction, hence neural networks have no notion of sequences or time. Recurrent Neural Networks (RNNs) perform predictions on input while using information derived from previous input. RNNs achieve this by adding a loop in the network. This loop allows the network to process sequential data while keeping track of an internal state in the hidden layers. The output from the activation function is sent both to the next layer (similar to a simple feed-forward neural network) and to the next iteration of the RNN. Therefore, RNNs are suitable when operating on sequences, for example, in time-series forecasting. An illustration of a simple recurrent neural network is found in figure 2.4

RNNs can be viewed as a sequence of neural networks that are trained in a sequence with backpropagation. The weights of an RNN are updated by using backpropagation through time (BPTT) [16]. In BPTT, the RNN is unfolded so that it becomes a feed-forward neural network with multiple layers and normal backpropagation is performed. However, if the number of layers becomes too large when unrolling the RNN, the calculated gradient can become too small. This is called the "vanishing gradient" problem, which occurs when the

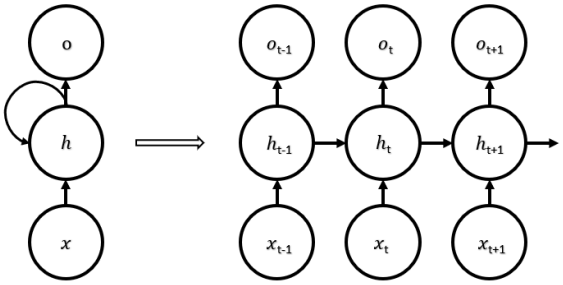

Figure 2.4: Example of a Recurrent Neural Network (left) and the unrolling of the network (right), where x is the input, o is the output and h is a neuron.

RNN operates on longer sequences. In practice, RNNs are not able to handle long-term dependencies [17].

2.4.5 Long Short Term Memory Networks

The Long Short Term Memory Networks (LSTM), introduced in 1997 [18], is a variant of the recurrent neural network capable of learning long-term dependencies. Both LSTMs and RNNs pass data in a sequential manner. However, an LSTM uses a gate mechanism to decide if it should either keep or forget information, allowing it to learn longer dependencies between input. The cell of an LSTM consists of a cell state and three gates: the forget gate, the input gate, and the output gate. The forget gate decides which information should be kept and which information should be discarded. The forget gate consists of a sigmoid layer that takes the last hidden state h_{t-1} and the current input x_t as input and outputs a value between zero and one, indicating how much of h_{t-1} should be kept. The input gate updates the cell state by using a tanh layer that creates new candidate values that are added to the existing cell state. The output gate decides what the next state should be. This is done by using a sigmoid layer that decides which parts of the cell state should be kept and multiplying the results with the cell state that has been put through a tanh function. When and what information to store, delete or update is learned by backpropagation and adjusting the weights of the gates via gradient descent. An illustration of the LSTM cell is shown in figure 2.5

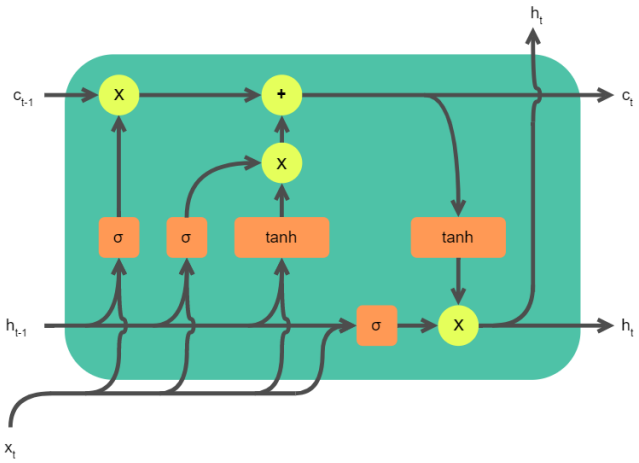

Figure 2.5: Internal structure of the LSTM (Long Short-term Memory) cell. Illustration by Guillaume Chevalier, 2018.

The standard LSTM network only considers past information when making a prediction. In many applications, such as time-series forecasting, the standard LSTM is sufficient due to the nature of the task. However, when dealing with textual data, where one has access to the whole sequence at time t, both past and future context information can be considered when making a prediction. In recent years, the Bidirectional Long Short Term Memory Network (BiLSTM) has gained popularity due to its ability to operate on sequential data such as text sequences and outperform unidirectional LSTMs [19, 20]. In simple terms, BiLSTMs are two LSTMs put together, where one network works in one direction (e.g., left to right in a text sequence) while the other works in the opposite direction. This allows the network to gain a richer context of the sequence at every time step.

2.5 Conditional Random Fields

Conditional Random Fields (CRF) are probabilistic graphical models often used in sequence labeling tasks that encode a conditional probability distribution used to predict the corresponding labels to a given input sequence [21]. For a given sequence observation $\boldsymbol{X}$, the objective is to find the probability of a sequence of labels $\boldsymbol{y}$. A linear chain structured CRF defines the conditional probability $P(\boldsymbol{y}|\boldsymbol{X})$ as follows:

$$P(\boldsymbol{y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \exp\Big(\sum_{k=1}^{N} U(\boldsymbol{x}_k, y_k) + \sum_{k=1}^{N-1} T(y_k, y_{k+1}) \Big) \tag{2.8}$$

where $T(y_k, y_{k+1})$ is a learnable transition matrix representing the transition probability of going from label k to label $k+1$, the $U(\boldsymbol{x}_k, y_k)$ are the emission scores representing how likely y_k is given the input $\boldsymbol{x}_k$, where each emission score has a associated learnable weight, and $Z(\boldsymbol{X})$ is a partition function used as an normalization factor in order to derive a probability. This normalization factor is computed using the *forward algorithm* or *backwards algorithm*. The labels that maximize $P(y_k|\boldsymbol{X})$ at each time step is taken to form a sequence of predicted labels, which can be done using the *Viterbi algorithm*. Because the probability $P(\boldsymbol{y}|\boldsymbol{X})$ is to be maximized, The CRF's weights are approximated using backpropagation with the the negative log likelihood, $-\log(P(\boldsymbol{y}|\boldsymbol{X}))$ as the loss function.

2.6 Multi-task Learning

In Machine Learning, the traditional way to approach a task is to train a single model to optimize a specific loss function. However, several tasks are often related to each other, thus, sharing commonalities and providing valuable information. By sharing a domain-specific representation between related tasks, a model can utilize the training signals from one task to train on another. This approach can be observed in human learning. For example, when a human learns a new task, he/she will often apply the knowledge gained from learning related tasks. In the domain of Machine Learning, Multi-task learning (MTL) is an approach to solve several tasks in parallel using a shared representation of common input, thus optimizing more than one loss function. Using the training signals of one task as an inductive bias for the other task can improve the generalization and robustness of the model. The purpose is to improve performance on a set of primary tasks through the inductive bias introduced by additional tasks. The primary tasks are referred to as target tasks, and the additional tasks are referred to as auxiliary tasks [22]. Multi-task learning has been used successfully across several NLP applications, including sequence labeling tasks such as Part-of-speech (POS) tagging, NER and syntactic chunking [23, 24, 25].

2.6.1 Model Architecture

In the context of NLP, multi-task learning is typically divided into two architectures: hard or soft parameter sharing of hidden layers. As shown in the right architecture in figure 2.6, hard parameter sharing is when tasks share full network layers between each other, while having separate task-specific output layers. This setting reduces the risk of overfitting. While hard parameter sharing is useful in many scenarios and often the norm, it can break down if tasks are not closely related [25]. In Soft parameter sharing (left architecture in figure 2.6), each task has its own separate network layer. These separate layers are then regularized during training to reduce the differences between the layers, encouraging layers to have similar weights.

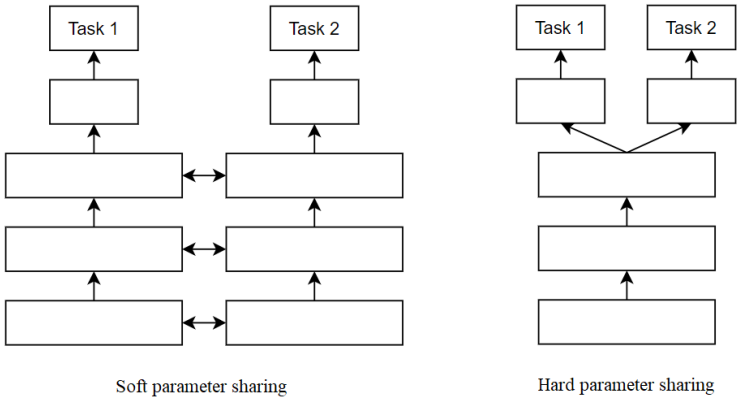

Figure 2.6: Example of two common multi-task learning model architectures.

Recent work [25, 26] introduces a hierarchical inductive bias between tasks, where low-level tasks (i.e tasks that are assumed to require less language understanding) are placed at the bottom layers and high-level tasks placed at the higher layers. An example of a hierarchical MTL approach can be seen in figure 2.7.

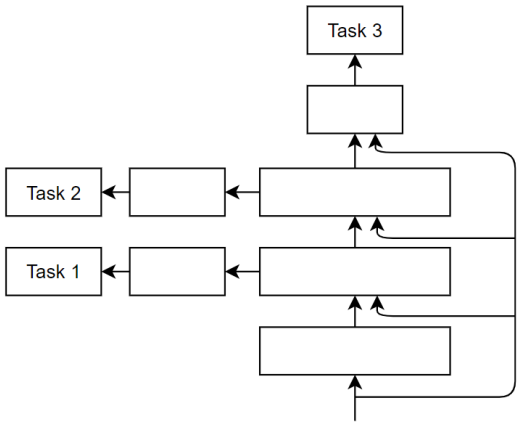

Figure 2.7: Example of a hierarchical multi-task learning model architecture.

In this example, task 1 could be a NER task, task 2 an Entity Mention Detection task, and task 3 a Coreference Resolution task. The tasks are regarded as increasing in complexity, where task 1 is regarded as the simplest task, and task 3 is regarded as the most complex.

2.6.2 Training Methodology

Up to this point in time, there does not exist a consensus on how to correctly train multi-task models [25]. One method of training a multi-task model is to consider a joint loss function, where the sum of the task-specific losses are calculated and minimized during training. Note that the task-specific losses are weighted uniformly in this training methodology, which can be counterproductive in settings where the goal is to train a model on a target task using auxiliary tasks. In that case, weights can be introduced to bias the model to devote more attention on minimizing the target-task's loss, as shown in equation 2.9.

$$\mathcal{L}_{sum} = \sum_{t}^{|T|} \lambda_t \mathcal{L}_t \tag{2.9}$$

where λ_t is the task-specific loss for task t. Another method for training a multi-task model is to choose a task at random at set intervals, calculate the task-specific loss and update the model parameters [25]. The random sampling of tasks can both be done uniformly or proportionally to the dataset sizes for each task.

2.7 Word Embeddings

The process of understanding and manipulating language can be highly complex. Language is full of abstractions and ambiguity, making it difficult for computers to process. In general, the understanding of language is learned through mathematical models that operate on numerical representations of the textual data. The process of converting textual data into numerical representations is commonly referred to as embedding. This is achieved by a function that maps, e.g., words to vectors of numbers to form a vector space. The vector space can, among other things, be used to measure relations between words or used as features in a machine learning classifier. The use of word embeddings was first proposed by Mikolov [27], who used a neural network to train d-dimensional word vectors used for predicting the next word in a sequence, a task that is formally known as language modeling. The d-dimensional word vectors thus provide a semantic meaning of a word.

2.7.1 Contextualized Word Embeddings

One drawback of word embeddings is that a word will always have the same numerical representation regardless of the context in which the word occurs. There are several cases where words can have different semantics and meaning depending on the context. For example, the word "bank" can refer to a financial institution or the land alongside a river. Knowing which semantic meaning the word refers to requires taking the surrounding context of the word into account.

Contextualized Word Embeddings are word embeddings that are mapped using both the word and the context surrounding the word, resulting in different embeddings of the word, depending on the context in which it occurs. These embeddings are a result from general-purpose language models trained on a large amount of text, such as Wikipedia articles and news articles. The training process of these language models often consists of next sentence prediction (NSP) or masked language modeling (MLM). In NSP, the objective is to determine, given a pair of sentences, if the second sentence comes after the first one, while in MLM, the objective is to predict words that have been randomly masked in a sequence. The tasks enable the models to learn both semantic and structural relationships between words, resulting in models that have a general understanding of natural language.

2.7.2 BERT

Bidirectional Encoding Representations from Transformers, often referred to as *BERT*, is a general-purpose language model based on the Transformer architecture, introduced in 2018 [28]. The Transformer architecture, which is a sequence to sequence model, consists of two separate mechanisms: the encoder and decoder. The encoder aims to encode the input sequence in a way that allows the model to understand context and temporal dependencies across the sequence, while the decoder decodes the encoded sequence to generate an output. Since BERT's goal is to generate a language model, only the encoder mechanism from the Transformer architecture is used. An illustration of the Transformer architecture is shown in figure 2.8, where the architecture on the left represents the encoder architecture.

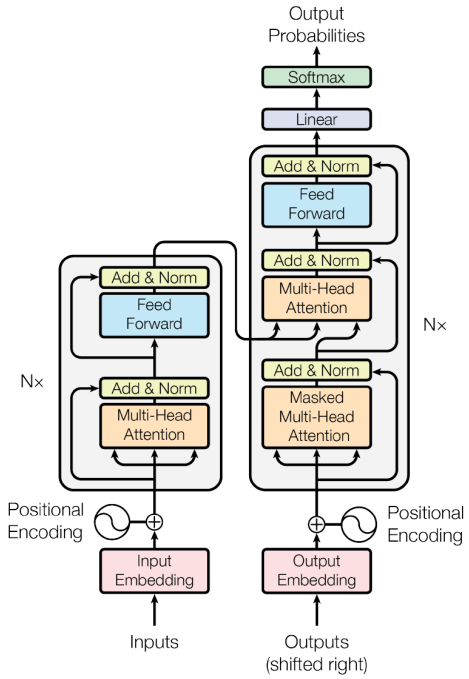

Figure 2.8: Architecture of the Transformer model. Illustration by Vaswani et al. [4], 2017.

The encoder component consists of multiple encoders stacked on top of each other. Each encoder layer consists of a self-attention layer and a feed-forward neural network that sends the output to the next encoder layer.

2.7.2.1 Self-attention

The self-attention layer consists of an attention mechanism that allow inputs to interact with each other, calculating how much *attention* each input should pay to the other inputs. Each input goes through linear projections to create three vector representations called *key*, *query* and *value*. A score is calculated by taking the scalar product between a query and key vector, representing how much focus should be placed on other parts of the input when encoding a certain input. This is done for each input in the input sequence. The resulting score is divided by the square root of the dimension of the key vectors and passed through a softmax operation in order to normalize the value. Each value vector is multiplied with the softmaxed

score in order to filter out irrelevant inputs. Lastly, the weighted value vectors are summed to produce the output of the self-attention layer. The formula for self-attention is as follows:

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.10}$$

In BERT, calculating self-attention is done across several attention heads called *multi-head attention* [28]. The idea is that each attention head has its own set of *key*, *query* and *value* vectors, resulting in different projections in order to calculate different types of relations and attentions. The output of each attention head is concatenated with each other to produce the final output to the feed-forward neural network.

2.7.2.2 Input Representation

Each input to BERT is a combination of three embeddings: *segment embeddings*, *token embeddings* and *positional embeddings*. The segment embeddings represent which sentence the input belongs to. The token embeddings are WordPiece embeddings [29] learned for each token in the token vocabulary. The positional embeddings express the position of the input in a sequence. The resulting input representation is a summation of all the previous mentioned embeddings. As shown in figure 2.9, a special token ([CLS]) is added to the first position in each sequence. This token is used as a sequence representation for down-stream classification tasks. Moreover, a separator token ([SEP]) is added between each sentence in order to differentiate between sentences.

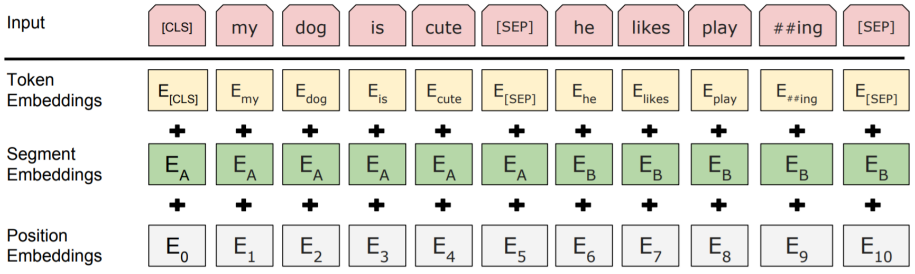

Figure 2.9: BERT input representation. Illustration by Devlin et al. [28], 2019.

2.7.2.3 Training BERT

What separated BERT from the previously introduced language models was that it was trained in a bidirectional sense, while the other models used traditional left-to-right or right-to-left pre-training [28]. BERT was trained both on the *next sentence prediction* and *masked language modeling* task, using data from the BooksCorpus [1] and the English Wikipedia, each containing 800 million and 2500 million words respectively. BERT was shown to outperform several state-of-the-art models when fine-tuned on tasks such as question answering and language inference. Moreover, it performed on par with other state-of-the-art models when fine-tuned on the CoNLL-2003 NER task [28].

2.8 Salient Named Entity Extraction

Salient entity detection, also referred to as document aboutness [3] or salient entity discovery, is the task of determining the *salience* of each entity in a document. Identifying salient entities

[1] https://www.english-corpora.org/googlebooks/

in a text document is similar to extracting terms representing the most salient information in a text document, i.e., keyword extraction [30]. The apparent difference between the two tasks is that the former is entity-centric. Gamon et al. [31] found that on average, 66 entities are mentioned on web pages, yet fewer than 5% of these entities are salient to the content. In salient named entity extraction, the task is to extract the salient *named* entities in a document. Thus, the task only focuses on determining the salience of named entities, as opposed to entities of any kind. Identifying salient named entities in text documents can help in the construction of knowledge bases and search engines, or provide valuable information for other NLP tasks. For example, Wu et al. [32] proposed a topic model named *Salient Entity Topic Model* that used salient entities to represent documents to learn rich topic distributions for entities and documents.

2.8.1 Related Work in Salient Entity Discovery

Dunietz and Gillick [30] used a two-stage approach to find salient entities in news articles. The approach consisted of first finding entity mentions using several NLP parsers such as a part-of-speech tagger, dependency parser, co-reference resolver and an entity resolver and then determining if each found entity is salient by using a regularized binary logistic regression model. The logistic regression model used features derived from the text document, such as the mention frequency of the entity and the position of the entity's first mention, together with a feature representing a notion of centrality derived from the PageRank algorithm. The model was compared to a positional baseline that classifies an entity as salient if it is mentioned in the first sentence of the document and showed to outperform the baseline's f1-score by 34%. The authors conclude that the addition of the centrality feature did not produce any substantial improvement over only using features derived from the text. However, using a co-reference system to derive the conjunction of the number of named, nominal and total mentions of an entity did show to add meaningful information to the model.

Trani et al.[3] proposed an Entity Linker which also predicts the saliency of the linked entities. As in [30], the process involved a two-stage process, consisting of a *Candidate pruning step* and a *Salience linking step*. The *Salience linking step* used a Gradient Boosted Decision Tree (GBDT) utilizing 22 features, grouped into either *light* or *heavy* features, depending on how computationally expensive the features were. The *light* features comprised of textual features derived from the text, such as the features used in [30]. The *heavy* features were computed from the graphs induced by Wikipedia links from the Wikipedia knowledge base, designed to model the relationship between entities in the text document. The authors found that the performance of the GBDT when only using the *light* features was similar to the performance when using the full set of the 22 features.

2.8.1.1 Keyword Extraction

A keyword can be defined as a term that provides a compact representation of the essential content of a document. Keywords are widely used in information retrieval systems and can help users to search specific topics and find related material. The task of automatically extracting keywords from a document is referred as *keyword extraction*, and has been widely studied. The problem of extracting only the salient named entities from a text document is similar to extracting terms that represent the most salient information contained in a text document, i.e., keyword extraction. The apparent difference between the two tasks is that only named entities are considered in salient named entity extraction.

There exists several approaches to keyword extraction, ranging from simple methods that rely on word count occurrences, to methods based on the graph-based syntactic representation of text, to the recent proposed approaches of using supervised deep learning to exploit the semantics of the text. There are mainly two approaches taken in modern deep learning methods for keyword extraction: sequence labeling and sequence generation. The sequence

labeling approach models the task as determining if each word or phrase in a sequence is a keyword or not, while the sequence generation approach generates a sequence of keywords given an input sequence. Recent work shows the success of Bidirectional Long short-term Memory networks paired with a Conditional Random Field employed on the keyword extraction task, which significantly outperformed the statistical baseline models [33, 34]. Martinc et al. [35] used a transformer-based model to perform keyword extraction using sequence labeling and achieved state-of-the-art performance. The authors also showed that their sequence labeling approach outperforms the sequence generation approach when evaluated on text documents that have the ground truth keywords in the body text.

2.8.2 Existing Entity Salience Datasets

The task of identifying salient entities in documents is a fairly new research area, resulting in a dearth of resources for training and evaluating approaches. Manually annotating datasets is highly time-consuming; thus, previous research often resorts to creating training automatically. Dunietz and Gillick [30] generated salience labels automatically for the existing *New York Times corpus* [36]. The annotations were constructed with the assumption that salient entities are mentioned in the article's abstract. Thus, all entities found in the abstract were used as the ground truth salient entities for an article. The resulting annotations were released publicly. However, due to copyright restrictions, the actual textual content of the articles was not released. Wu et al. [37] used Wikinews category tags to construct a Wikinews corpus containing the salient entities for each article, with the assumption that the entities found in the category tags for a news story are salient. Another entity salience detection dataset is the *Reuters-128 Salience* dataset, created by Dojchinovski et al. [38] . To obtain gold standard entity salience annotations, the authors used a crowdsourcing tool to collect annotations from judges. Three judgments from different annotators were collected and aggregated to reach a final salient annotation for each entity.

3 Method

The following chapter describes the methodology taken to answer the research questions presented in the introduction chapter. First, the task of salient named entity extraction is formally stated. Second, the method taken to automatically create the training data is described. Lastly, the studied approaches are presented, together with their corresponding implementation details.

3.1 Task Formulation

An entity's salience is subjective to the reader's perspective and can be challenging to define. The salience can be assessed both from the interests of readers and from the entity's prior importance outside of the document. For example, although *Donald Trump* is an important entity in general, he can be peripheral to the news article's underlying news. This study assumes that the source of salience is local to the news article and that no a priori information is taken into consideration, nor the intent of the reader. The task of Salient Named Entity Extraction is formulated as follows:

> Given a news article, extract all named entity mentions of type Organization that are believed to be central to the news.

3.2 Dataset

To be able to train and evaluate the supervised models, suitable annotated training data needs to be present. It is challenging to find an appropriate dataset for the studied problem due to the limited research done in the area. As presented in 2.8.2, there exist training and evaluation sets for the entity salience task. However, these datasets are intended for the task of finding salient entities, and not only named entities, as is the case for this thesis. The following section describes the methodology taken when creating the dataset used for training and evaluating the different approaches explored in this thesis. Moreover, general statistics of the resulting dataset is presented.

3.2.1 Creating the training corpus

In this study, an approach similar to Dunietz and Gillick [30] is taken. A dataset serving as training data is constructed by using an existing financial news corpus that includes each news article's headline. The article headline's named entities are used as the ground truth salient named entities for the news article. A headline for a given text document can be seen as a compact summarization of the the central meaning of the text. The taken approach assumes that the named entities mentioned in an article's headline are highly relevant to the news, and all other named entities found in the body text can be disregarded.

In order to extract named entities for the article's headline and body text, the off-the-shelf named entity linking system Radboud Entity Linker (REL) [39] is used. REL consists of three components: mention detection, candidate selection and entity disambiguation. The mention detection is performed using Flair [40], a NER-tagger with a BiLSTM-CRF architecture operating on contextualized string embeddings, which achieved state-of-the-art performance on the CoNLL 2003 NER task in 2018. The REL system uses a dump of the 2019 English Wikipedia as its knowledge base to link named entity mentions to their corresponding unique Wikipedia page. The reason why a NEL system is used, instead of only using a NER-tagger is to make it possible to consider name variations of a named entity in the evaluation phase. For example, *Volkswagen* and *VW* refer to the same car manufacturing company. If *Volkswagen* is in the set of ground truth salient entities, and if a model were to predict that *VW* is a salient entity, the prediction should be considered accurate. The NEL system can be used to identify that *Volkswagen* and *VW* refer to the same unique Wikipedia page, and therefore refer to the same company. The entity linker is first employed on the headline of each news article and then on the body text. The span and unique Wikipedia page identifier is saved for each corresponding named entity mention of the Organization type (ORG).

There is a risk that the mention disambiguation phase in the REL fails to find the corresponding Wikipedia page for a named entity, which will result in the REL missing to tag the mention. In order to reduce the risk of failing to recognize a company mention, the Flair NER model is separately employed on the body text to find mentions that the REL system failed to capture. However, the corresponding Wikipedia pages for these mentions cannot be obtained.

The process of annotating the training set can be summarized as follows: For each news article in the dataset:

1. Perform NEL on the article's headline and save the span and Wikipedia page for each found named entity. Next, perform NER on the headline and save the span of named entity mentions that were not captured by the NEL system. If no named entities were found in the headline, disregard the article.

2. Perform NEL on the article's body text and save the spans and Wikipedia page for each found named entity Next, perform NER on the body text and save the span of named entity mentions that were not captured by the NEL system. If a named entity has the same Wikipedia page or spelling as one of the named entities in the headline, it will be marked as salient. If no named entities were found in the body text, disregard the article.

A portion of news articles are manually annotated to derive the salient named entities to form a validation and test set. When manually annotating the validation and test set, news articles containing headlines such as *"Three major US banks fined for anti-money-laundering violations"*, i.e., articles that do not mention any named entities in the headline, are also considered. The training set, together with the human-annotated validation set is used when training the different approaches, where the validation set is used to evaluate the approaches during training to ensure that they do not overfit on the training set. When the development phase is done, the three approaches are evaluated on the test set.

3.2.2 Dataset Characteristics

The dataset used in this study consists of financial news articles from Reuters News spanning over a time period of two years, from early 2018 to early 2020. The topics of the news articles are typically related to macroeconomics, mergers and acquisitions activity, CEO changes and corporate wrongdoing. News articles that only report stock movements, i.e. listing how different stocks have behaved throughout the day, are excluded from the dataset. Furthermore, news articles that only mention one named entity in total are also excluded from the dataset, as well as news articles that have a word length longer than 500. The statistics of the train, validation and test set are presented in table 3.1.

Dataset characteristic	Train	Val	Test
# of articles	9,428	157	150
Average # unique salient named entities per article	1.7*	2.9	2.8
Average # salient named entity mentions per article	5.2*	6.7	6.5
Average # named entity mentions per article	10.9*	10.7	10.6
Average # tokens per article	235.0	202.6	210.3

Table 3.1: General statistics of the processed datasets. * denotes statistics that are estimated using the NER and NEL systems.

It is important to note that the validation and test set are human-annotated, while the train set is automatically annotated. Thus, the statistics for the train set will not be fully accurate, due to false negatives and false positives error from the off-the-shelf NER and NEL systems. Table 3.2 presents the Flair NER-tagger's performance when detecting named entities of type Organisation on the test set. Table 3.3 presents the performance of the taken approach for building the training set when evaluated on the human-annotated test set. The high precision in table 3.3 indicates that the named entities found in the headline are in most cases salient. However, the named entities found in the headline only make up for around 57% of the ground truth salient entities, which is shown by the recall being 0.572.

Precision	Recall	F1-score
0.944	0.920	0.932

Table 3.2: The precision/recall/f1-score of the Flair NER-tagger on the task of identifying named entities of type Organization, evaluated on the human-annotated test set.

Precision	Recall	F1-score
0.918	0.572	0.705

Table 3.3: The precision/recall/f1-score of the taken training set generation approach on the human-annotated test set.

3.3 Approach 1: Two-stage Approach

The first approach implemented is the two-stage approach using a NER tagger, NEL system, knowledge base and a binary classifier utilizing hand-crafted features. The two-stage approach consists of first identifying named entity mentions in the body text to form a list of candidate named entities. The candidate list is then pruned using a binary classifier, for which each named entity is classified to be salient or not salient. An overview of the two-stage approach is shown in figure 3.1.

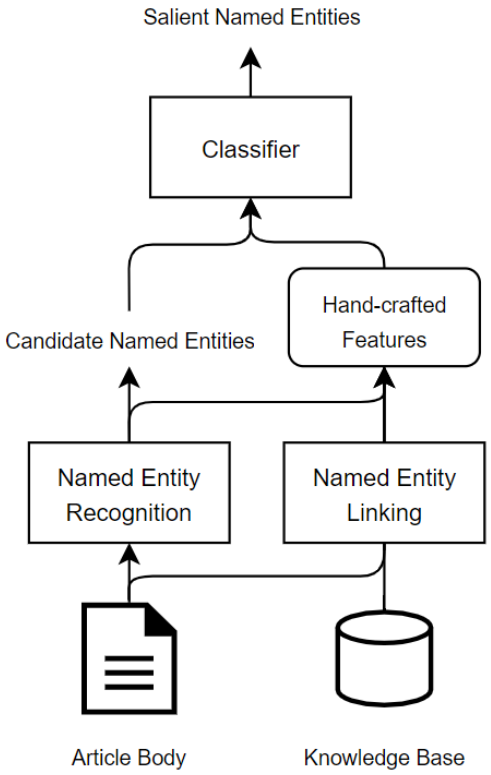

Figure 3.1: Overview of the two-stage approach using Named Entity Recognition and a classifier with hand-crafted features.

The off-the-shelf Flair model is used to perform NER on the article's body text. In this stage, only named entities that have the IOB tags: *B-ORG*, *I-ORG* are saved to the candidate list. Next, hand-crafted features are constructed for each candidate named entity. In this stage, the 2019 Wikipedia dump is used together with the Radboud Entity Linker to perform mention disambiguation to derive features to be used by the binary classifier. For each candidate named entity, the classifier uses the corresponding features to determine if the named entity is salient or not.

3.3.1 Feature Selection

The salience classifier uses extracted features from the body text, together with a knowledge base to derive the salience of a named entity in the candidate list. The hand-crafted features evaluated in this study are shown in table 3.4.

The proposed hand-crafted features are inspired from the work in [30], [38] and [3]. The *first-mention* feature is based on the hypothesis that salient mentions are often mentioned early in the article. The *first-mention* feature for a named entity is the index of the sentence where the named entity, or the unique identifier for said named entity, is first mentioned in. The *entity-frequency* feature is based on the hypothesis that salient named entities are often mentioned multiple times in the article, thus having knowledge of the number of times the entity is mentioned can provide meaningful information to the classifier. The *KB-entity-frequency* feature is designed to handle cases where there exists several name variations for a named entity. The *entity-frequency* cannot always capture name variations when it counts the number of mentions in the article. The *KB-entity-frequency* is derived by a named entity linker which has the ability to map both *Volkswagen* and *VW* to the same unique Wikipedia

Feature	Description
first-mention	Index of the sentence in which the first mention of the named entity appears.
entity-frequency	Number of times the head word of the named entity appears.
KB-entity-frequency	Number of times the knowledge base unique identifier of the named entity appears.
unique-entity-frequency	Total number of unique named entities mentions.
unique-KB-entity-frequency	Total number of unique knowledge base identifiers.
sentence-count	Number of sentences in the article.

Table 3.4: The hand-crafted features used by the classifiers in the two-stage approach.

identifier, thus it represents the number of mentions in the article to the unique Wikipedia identifier that the named entity is associated with.

Figure 3.2 presents a news article example, where the identified named entities are highlighted in bold. Table 3.5 show the extracted features for each identified named entity. Notice that the *first-mention* value for *Google* is 1, even though it is first mentioned in the second sentence. This is because *Google Inc.* (which has the same unique knowledge base identifier as *Google*) is mentioned in the first sentence.

> Search and advertising giant **Google Inc.** closed its deal to buy fitness tracking company **Fitbit**, the companies said on Thursday, even as U.S. and Australian competition regulators said they were continuing probes of the $2.1 billion transaction. The **Justice Department**, which sued **Google** in October for allegedly violating antitrust law in its search and search advertising businesses, said it "has not reached a final decision about whether to pursue an enforcement action" regarding the **Fitbit** deal.

Figure 3.2: News article example

Named entity	FM	EF	KB-EF	unique-EF	unique-KB-EF	SC	KB identifier
Google Inc.	1	2	2	4	3	2	wiki/Google
Google	1	2	2	4	3	2	wiki/Google
Justice Department	2	1	1	4	3	2	N/A
Fitbit	1	2	2	4	3	2	wiki/Fitbit

Table 3.5: Extracted features for each identified named entity. *FM* denotes *first-mention*, *EF* denotes *entity-frequency*, *KB-EF* denotes *KB-entity-frequency*, *unique-EF* denotes *unique-entity-frequency*, *unique-KB-EF* denotes *unique-KB-entity-frequency*, *SC* denotes *sentence-count*, *KB identifier* denotes the unique knowledge base identifier derived from the Radford Entity Linker using the 2019 Wikipedia knowledge base.

The impact each feature has on the binary classifier's accuracy is derived by iteratively adding features to the model and documenting the possible gain in precision, recall and f1-score. The first feature added to the model is the feature resulting in the highest f1-score when only one feature is used. Moreover, the f1-score will be prioritized as the evaluation metric when performing feature selection.

3.3.2 Model Selection

Similar to [3], a Gradient Boosting Decision Tree and a Random Forest classifier are considered for model selection. The models are evaluated using the precision, recall and f1-score metrics and compared against a positional baseline classifier that classifies a named entity as salient if it is mentioned in the first sentence of the article. Hyperparameter tuning using grid search is performed on the classifiers before they are compared against each other. The hyperparameters considered for tuning in the Random Forest classifier are *n_estimators* and *max_depth*, where n_estimators represent the total number of trees in the forest, while max_depth denotes the maximum depth of the trees. For the Gradient Boosting Decision Tree, the hyperparameters: *n_estimators*, *max_depth* and *learning_rate* are considered for tuning. Furthermore, the *decision_boundary* for both models are also tuned.

3.3.3 Implementation Details

The classifiers are trained and evaluated in Python 3.6 using the popular machine learning libraries Scikit-learn [1] and XGBoost [2]. The scikit-learn library provides various machine learning algorithms and utilities, which is used in this thesis to build the Random Forest classification model and calculate performance metrics. The XGBoost library is used to build the Gradient Boosting Decision Tree. XGBoost is an open-source gradient boosting library designed to be both time and memory-efficient by performing parallelization and cache optimization.

3.4 Approach 2: Single-task Sequence Labeling

The second approach explored in this thesis treats the task of identifying salient named entities in a news article as a sequence labeling task, where the full text article represents the sequence. Each token/word in the document is labeled as either being part of a salient named entity mention or not. The labeling schema used is the IOB2-format with the following labels:

1. **B-SNE**: indicates that the word is the beginning of a salient named entity.
2. **I-SNE**: indicates that the word is part of a salient named entity. The I-SNE tag is only used when the preceding tag is B-SNE.
3. **O**: indicates that the word is not part of a salient named entity.

The sequence labeling SNEE task can be formally stated as:

> let d = $\{w_1, ..., w_n\}$ be a document, where w_i represents the i$^{\text{th}}$ token in d. Assign one of three labels y = $\{B\text{-}SNE, I\text{-}SNE, O\}$ to each w_i in d.

3.4.1 Single-task Model Architectures

When building the sequence labeling model, two different model architectures are considered. The first model architecture, henceforth referred to as *BERT-finetuned* consists of a pre-trained BERT model, where a simple linear layer is added to handle the output of BERT. The output of the linear layer is fed through a softmax layer, i.e. the normalized exponential function, to obtain a probability distribution over the label space. The parameters of BERT, together with the parameters of the linear layer are jointly trained. The second model architecture, hereafter referred to as *BiLSTM-based*, introduces a BiLSTM between the BERT layer and the linear layer, where the BiLSTM uses the last hidden layer from BERT as input features. When training the BiLSTM-based model, the parameters of the pre-trained BERT layer

[1] https://scikit-learn.org/stable/
[2] https://xgboost.readthedocs.io/en/latest/index.html

are not fine-tuned. The comparison of the two model architectures is similar to the comparison made in BERT's original paper, where the authors compared fine-tuning BERT against the approach of using the output of BERT as contextual embeddings to a BiLSTM layer [28].

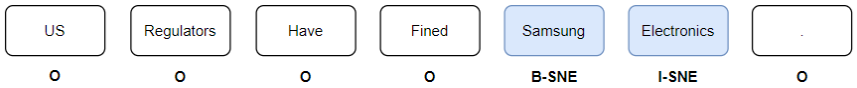

Figure 3.3: Single-task sequence labeling example with SNEE labels for each token in the sequence.

Furthermore, experiments with a CRF layer are conducted, where the CRF layer is added to the last layer in both model architecture variations. For the models that use a CRF as the final layer, the Viterbi algorithm is used to find the most likely label sequence to derive a predicted label sequence. Moreover, a simple CRF model is also trained and evaluated to act as a sequence labeling baseline model. The features used by the CRF baseline are shown in table 3.7. To summarize, the evaluated neural model architectures are shown in table 3.6.

Model Name	**Layers**
BERT-finetuned	BERT + linear
BERT-finetuned + CRF	BERT + linear + CRF
BiLSTM-based	BERT + BiLSTM + linear
BiLSTM-based + CRF	BERT + BiLSTM + linear + CRF

Table 3.6: Variations of neural model architectures considered for single-task sequence labeling.

Feature Name	**Description**
word.index	Index of word in document
word.lower	The lowercased word
word.isupper	If all characters in the word are uppercase
word.istitle	If first character in word is uppercase
word.isdigit	If word is a digit
word.prefix-1	First character in word
word.prefix-2	First two characters in word
word.prefix-3	First three characters in word
word.suffix-1	Last character in word
word.suffix-2	Last two characters in word
word.suffix-3	Last three characters in word
-3:word.lower	The lowercased word three indices before
-3:word.isupper	If all characters in the word three indices before is all uppercase
-3:word.istitle	If first character in the word three indices before is uppercase
...	
+3:word.lower	The lowercased word three indices after
+3:word.isupper	If all characters in the word three indices after is all uppercase
+3:word.istitle	If first character in the word three indices after is uppercase

Table 3.7: Features used by the Conditional Random Field baseline sequence labeling model. The features are derived for each word in the text document.

3.4.1.1 Implementation Details

All model architecture variations are implemented using the PyTorch[3] and the Transformers[4] library. The implementation of BERT-finetuned is inspired by the architecture used in [28] when experimenting on the NER task. The BERT layer has a 0.4 dropout probability for all fully connected layers. The linear layer in BERT-finetuned takes the hidden outputs of the BERT layer, thus it takes 768 features as its input and outputs 3 features. During training, the output of the linear layer is fed into a cross entropy loss function and the loss is calculated. The BiLSTM layer in the BiLSTM-based model has two layers, an input size of 768, a hidden size of 256 and uses a dropout probability of 0.4 on each output layer. The outputs of the BiLSTM are fed to the linear layer.

All model architectures use the Adam optimizer with weight decay for the optimization algorithm, with a learning rate = 1e-5 and a weight decay = 1e-3. The learning rate has a linear warmup phase of 3000 steps and then decreases linearly to zero. Early stopping is implemented and the training is stopped when no further improvements on the validation set in terms of f1-score are observed, with a patience of 500 batches. Since sequence labeling is done on a document level, i.e., the models take the whole article text as input, the batch size is set to 4 to avoid CPU memory problems. Moreover, the "bert base cased" version of BERT is used, which was trained on cased English text, has 12-layers, 768 hidden features and 12 attention heads, resulting in 109 million parameters.

3.5 Approach 3: Multi-task Sequence Labeling

In the third approach, a multi-task learning (MTL) methodology is taken to identify salient named entities in news articles. The motivation behind using MTL hail from such systems' improved ability to generalize. Having a model that generalize well is especially important in this study, due to the training data being automatically created and possibly not sharing the same label distribution as the validation and test set. Therefore, the model should learn general syntactical and relational representations while ignoring the data-dependent noise in order to avoid overfitting.

The fact that finding salient named entities in a document is tightly coupled with identifying named entity mentions, suggests that a multi-task learning approach, where a model trains on an auxiliary NER task and the SNEE task simultaneously, could outperform models solely trained on the SNEE task. Moreover, because the training data together with the labels are automatically generated using the Flair NER tagger, the resulting named entity labels can be used as training data for the auxiliary NER task without any additional label generation.

Figure 3.4: Multi-task sequence labeling example with SNEE and NER labels for each token in the sequence.

3.5.1 Multi-task Model Architectures

The same model architectures compared in the single-task approach are also compared in the MTL approach, namely the BERT-finetuned and BiLSTM-based models. However, for the

[3]https://pytorch.org/
[4]https://huggingface.co/transformers/

MTL approach, a hard parameter sharing architecture is implemented in both model variations, where a task-specific layer is introduced for the NER task. The type labels used in the NER task are *Organization*, *Person*, *Location* and *Misc*.

3.5.1.1 Implementation Details

The implementation details of the multi-task model architectures are similar to the single-task models. The main difference is that the loss used in the MTL models is calculated by summing the task specific losses, as shown in equation 2.9. The weights for the two task specific losses are set to 1. As in the single-task models, a linear layer and a CRF layer is used for the NER task.

3.6 Comparing The Approaches

Because approach two and three perform sequence labeling using the IOB-format while the first approach is a two-stage approach, where the final output is of the form of a binary classification for each found named entity, the different approaches cannot be directly compared to each other using their corresponding precision/recall and f1-scores. To give a fair evaluation and comparison between the approaches, each approach will produce a list of predicted unique named entity mentions for each news article. The task of producing a list of salient named entity mentions for each document is referred to as the Salient Named Entity Extraction (SNEE) task. In the two-stage approach, for each news article, the named entities that are classified as salient are added to the list of predictions. In sequence labeling approaches, the label output from the sequence labeling models are used to form a list of predictions. This is done by extracting the tokens that the model have labeled either *B-SNE* or *I-SNE* to form a list of words that are believed to be salient named entities.

Spelling variations and abbreviations of named entities often occur in news articles. For example, the author of a news article might refer to the company *Hennes & Mauritz* by its proper name in the first sentence of the body text, but use the abbreviation *H&M* later on in the article. Both mentions refer to the same company and should be treated equally. Therefore, name variation of the ground truth salient named entity are also considered in the evaluation phase. Thus, the goal of the approaches is to, in addition to identify the salient named entities in the text document, also identify all the name variations of the salient named entity as salient. The outputted set of predictions by each approach is compared to the ground truth set of salient named entities for each article.

3.6.1 Significance Test

Significance testing is performed on the results in order to substantiate the difference in f1-score between the best performing models and their corresponding baseline models. The following null hypothesis is used:

$$\begin{aligned} H_0 &: \text{No difference in f1-score between model } m \text{ and baseline } bl \\ H_1 &: \text{Model } m \text{ has a higher f1-score than baseline } bl \end{aligned} \tag{3.1}$$

The mean and variance of the difference between the f1-score is approximated using the delta method [41]. The delta method uses a one-step Taylor expansion to expand the function of a random variable about its mean, in turn estimating its variance. The test statistic therefore becomes:

$$T = \frac{g(\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m)}{\sqrt{\frac{1}{n} \nabla g|^T_{\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m} \hat{\Sigma} \nabla g|_{\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m}}} \sim N(0,1) \text{ given } H_0 \tag{3.2}$$

where

$$g(\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m) = f1(\hat{p}_m, \hat{r}_m) - f1(\hat{p}_{bl}, \hat{r}_{bl}) \tag{3.3}$$

i.e., the difference in f1-score between model m and the baseline bl, $\nabla g|_{\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m}$ is the first-order Taylor expansion of $g(\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m)$, $\hat{\Sigma}$ is the sample covariance matrix of $(\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m)$, $\hat{r}_i$ is the estimated recall for model i and $\hat{p}_i$ is the estimated precision for model i. Furthermore, the following test rule is used

$$\text{reject } H_0 \text{ if } T_{obs} \geqslant \Phi^{-1}(0.95) \tag{3.4}$$

where Φ is the cumulative standard normal density function. Hence, a significance level of 5% is used. The P-value is calculated by

$$\textit{P-value} = 1 - \Phi\left(\frac{g(\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m)}{\sqrt{\frac{1}{n} \nabla g|^T_{\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m} \hat{\Sigma} \nabla g|_{\hat{p}_{bl}, \hat{p}_m, \hat{r}_{bl}, \hat{r}_m}}}\right) \tag{3.5}$$

4 Results

The following chapter presents the results of the conducted experiments. First, the results from the feature selection phase in the two-stage approach are presented, together with the resulting performance on the SNEE task. Subsequently, the resulting performance of the sequence labeling approaches are presented. Lastly, the best performing models in each approach category are compared against each other on the SNEE task.

4.1 Two-stage Approach

The following section presents the results from the feature selection experiments and the classification model selection phase for the two-stage approach.

4.1.1 Feature Selection

Table 4.1 presents the validation precision/recall/f1-score on the *Salient* class when training a Random Forest classifier using only one feature at a time. The results show that the *first-mention* feature achieves the highest f1-score on the validation set (0.744 f1-score), slightly outperforming the *entity-frequency* feature (0.736). The *sentence-count* and *unique-KG-entity-frequency* features do not provide significant information to the classifier, with their f1-score being equal to the f1-score resulting from predicting that all named entities are salient.

Table 4.2 presents the precision/recall/f1-score on the validation set when the model features are incrementally added, starting with the *first-mention* feature. Note that hyperparameter tuning is performed in each iteration. The results show that the highest f1-score (0.782) is achieved when using the two features: *first-mention* and *entity-frequency*. When adding the *KB-entity-frequency*, the f1-score slightly decreases and continues to decrease when adding the *unique-entity-frequency* feature.

4.1.2 Model Results

Table 4.3 presents the test set precision/recall/f1-score on the *Salient* class when training the ensemble models using the combination of features that yielded the highest f1-score in the feature selection phase, namely *first-mention* and *entity-frequency*. The performance of the positional baseline is shown for comparison, as well as the resulting performance when always

Feature	Precision	Recall	F1
first-mention	0.679	0.823	**0.744**
entity-frequency	0.702	0.773	0.736
KB-entity-frequency	0.786	0.600	0.681
unique-entity-frequency	0.489	0.832	0.616
sentence-count	0.447	0.972	0.612
unique-KB-entity-frequency	0.441	1	0.612

Table 4.1: Validation set precision/recall/f1-score on the *Salient* class for the individual features listed in table 3.4.

Feature	Precision	Recall	F1
first-mention	0.679	0.823	0.744
+ entity-frequency	0.751	0.817	**0.782**
+ KB-entity-frequency	0.686	0.895	0.776
+ unique-entity-frequency	0.671	0.849	0.749
+ sentence-count	0.755	0.773	0.764
+ unique-KB-entity-frequency	0.750	0.762	0.756

Table 4.2: Validation set precision/recall/f1-score on the *Salient* class when iteratively adding the features listed in table 3.4.

predicting *Salient = True*. The model resulting in the highest f1-score is the *XGBoost* classifier with the hyper-parameters listed in table 4.4 with a corresponding f1-score of 0.767, which marginally outperformed the *Random Forest* classifier. The XGBoost classifier has a 10.4% relative increase in f1-score compared to the positional baseline, corresponding to a difference of 0.072 f1 points.

Model	Precision	Recall	F1
Random Forest	0.740	0.794	0.766
XGBoost	0.725	0.815	**0.767**
Positional baseline	0.785	0.623	0.695
Salient = True	0.443	1.000	0.614

Table 4.3: Test set precision/recall/f1-score on the *Salient* class.

Model	decision_boundary	n_estimators	max_depth	learning_rate
Random Forest	0.3	40	10	-
XGBoost	0.3	40	10	0.1

Table 4.4: Optimal hyperparameters for the binary classifiers in the two-stage approach.

Table 4.5 presents the macro-averaged precision/recall/f1-score per document on the SNEE task. The precision and recall is averaged over all documents and the f1-score is then calculated using the averaged precision and recall. The results show that using the *Random Forest* classifier in the two-stage approach results in the highest f1-score on the SNEE task, and has a 10.2% relative improvement over the positional baseline. Note that the recall for *Salient = True* is 0.982 and not 1. This is a result from the NER-tagger producing false negatives when identifying named entities.

4.2 Sequence Labeling Approach

The following section presents the experiment results of the sequence labeling approaches on the task of labeling salient entities and on the SNEE task. Table 4.6 presents the performance

Model	Precision	Recall	F1
Random Forest	0.765	0.838	**0.800**
XGBoost	0.752	0.853	0.799
Positional baseline	0.771	0.686	0.726
Salient = True	0.480	0.982	0.645

Table 4.5: Macro-averaged test set precision/recall/f1-score for each evaluated classifier in the two-stage approach on the SNEE task.

of the single-task sequence labeling models. The reported precision/recall/f1-score corresponds to the model's performance on the *SNE* label. In general, fine-tuning BERT achieves a higher f1-score compared to when using a BiLSTM layer and treating the output from BERT as input embeddings to the BiLSTM layer. The largest difference in f1-score between the two model architecture configurations is observed when a CRF is added to the last layer. In this case, the relative improvement in f1-score when fine-tuning BERT over using a BiLSTM is 2,3%. Furthermore, the results show that the f1-score generally improves when adding a CRF to the model architecture, although the increase is marginal in the BiLSTM-based architecture. The best performing sequence labeling model, BERT-finetuned + CRF, outperforms the CRF-baseline model by 0.17 F1-points, which constitutes an 25.7% relative improvement.

Model	Precision	Recall	F1
BERT-finetuned	0.829	0.820	0.824
BERT-finetuned + CRF	0.813	0.851	**0.831**
BiLSTM-based	0.826	0.795	0.810
BiLSTM-based + CRF	0.833	0.792	0.812
CRF-baseline	0.819	0.585	0.682

Table 4.6: Test set precision/recall/f1-score from the single-task sequence labeling models on the *SNE* label.

Table 4.7 presents a comparison between the sequence labeling models' performance in the different learning settings: *Single-task* and *Multi-task*. When examining the relative increase in f1-score observed when using a multi-task learning approach, the results varies. The multi-task learning setup increases the f1-score in the BERT-finetuned model architectures, but decreases the f1-score in the BiLSTM-based model architectures. The largest increase from usng a multi-task learning setup is observed for the BERT-finetuned model, which has a relative increase of 1.5% in f1-score over its single-task counterpart. On the other hand, the multi-task version of the *BiLSTM-CRF* model amounts t a 1.2% relative decrease in f1-score.

Table 4.8 presents the sequence labeling models' performance on the SNEE task. The results are similar to the results presented in table 4.7, and shows that the multi-task BERT-finetuned + CRF model produces the highest f1-score, outperforming the CRF baseline by 0.11 f1 points, constituting a 16% relative increase. The highest relative increase from using a multi-task learning setup is observed for the BERT-finetuned + CRF model, which has a relative increase of 3.1% in f1-score over its single-task counterpart. However, for the BiLSTM-based models, the largest relative decrease in f1-score amounts to -3.5%.

Lastly, table 4.9 presents a comparison between the multi-task Bert-finetuned + CRF and the Flair NER tagger's ability to identify named entities of type organization, i.e., their performance on the NER task. The table shows that the two models result in the same recall. However, the Flair NER tagger achieves a higher precision, which in turn yields a higher f1-score.

Type	Model	Precision	Recall	F1
Single-task	BERT-finetuned	0.829	0.820	0.824
Multi-task	BERT-finetuned	0.833	0.841	0.837
Single-task	BERT-finetuned + CRF	0.813	0.851	0.831
Multi-task	BERT-finetuned + CRF	0.839	0.839	**0.839**
Single-task	BiLSTM-based	0.826	0.795	0.810
Multi-task	BiLSTM-based	0.837	0.776	0.805
Single-task	BiLSTM-based + CRF	0.833	0.792	0.812
Multi-task	BiLSTM-based + CRF	0.795	0.811	0.803
Single-task	CRF-baseline	0.782	0.572	0.661

Table 4.7: Test set precision/recall/f1-score for the sequence labeling models on the *SNE* label.

Type	Model	Precision	Recall	F1
Single-task	BERT-finetuned	0.771	0.760	0.766
Multi-task	BERT-finetuned	0.770	0.789	0.779
Single-task	BERT-finetuned + CRF	0.740	0.794	0.766
Multi-task	BERT-finetuned + CRF	0.791	0.789	**0.790**
Single-task	BiLSTM-based	0.768	0.742	0.755
Multi-task	BiLSTM-based	0.773	0.728	0.750
Single-task	BiLSTM-based + CRF	0.778	0.751	0.764
Multi-task	BiLSTM-based + CRF	0.734	0.741	0.737
Single-task	CRF-baseline	0.742	0.630	0.681

Table 4.8: Macro-averaged test set precision/recall/f1-score from the sequence labeling models on the SNEE task.

Model	Precision	Recall	F1
Multi-task Bert-finetuned + CRF	0.908	0.920	0.914
Flair NER Tagger	0.944	0.920	**0.932**

Table 4.9: Comparison between the best performing multi-task sequence labeling model and the off-the-shelf Flair NER tagger on the NER task.

4.3 Comparison of Approaches

Table 4.10 presents a comparison between the three proposed approaches' best performing models on the SNEE task, together with their corresponding baseline models. The test set macro-averaged precision/recall/f1-score on the SNEE task is reported and the models are ranked in terms of their f1-score. The P-value for the null hypothesis (presented in section 3.6.1) is reported for each proposed model. The two-stage approach using a Random Forest classifier yields the highest f1-score among the approaches, outperforming the best single-task and multi-task sequence labeling models by 0.034 and 0.01 f1 points, respectively. The two-stage approach also yields the highest recall among the approaches, while still keeping a high precision. The P-value for each proposed model is below the significance level (0.05), which indicate that the 95%-significance test for each model rejects their respective null hypothesis and hence (in the meaning of f1-score) the model's fl-scores are statistically significant higher compared to their corresponding baseline.

Approach	Type	Model	Precision	Recall	F1	P-value
Two-stage	-	Random Forest	0.765	0.838	**0.800**	0.029
Sequence labeling	Multi-task	BERT-finetuned + CRF	0.791	0.789	0.790	0.003
Sequence labeling	Single-task	BERT-finetuned + CRF	0.740	0.794	0.766	0.007
Two-stage	-	Positional baseline	0.771	0.686	0.726	
Sequence labeling	Single-task	CRF-baseline	0.742	0.630	0.681	

Table 4.10: Comparison and ranking of the best performing models in their corresponding approach category on the SNEE task.

5 Discussion

The following chapter discusses the work conducted in this study. First, the results from the conducted experiments are discussed and analyzed. Second, the chosen methodology is reviewed, where possible weaknesses are identified and suggested improvements are presented. Third, the work is put in a wider context, discussing the ethical and societal aspects of the work.

5.1 Results

In the following section, the results of the conducted experiments are examined and discussed.

5.1.1 Features

The feature selection experiments showed that the optimal combination of features to use in a classifier for SNEE consisted of the two features *first-mention* and *entity-frequency*. The fact that knowing the total number of mentions of an entity and how early on in the article it is mentioned is important for determining salience, is consistent with earlier work in salience detection [30, 38, 31]. However, related work such as [38] reported that adding document-level features such as the *unique-entity-frequency* improves the f1-score, which was not the case in this study. The *unique-entity-frequency*, *unique-KB-entity-frequency* and *sentence-count* did not contribute to any improvements of the classifier in this study's experiments. The *KB-entity-frequency* feature, which counts the number of times the named entity's unique identifier is mentioned in the text, was introduced to aid in handling name variations of named entities. However, the feature was not shown to positively contribute to the model's performance. A possible explanation for this could be that the *entity-frequency* feature already incorporates the information gained by introducing the *KB-entity-frequency* feature. Recall that the *entity-frequency* feature counts the number of times the *head word* of the named entity appears in the text, meaning that it will be able to handle name variations when the head word of two name variations are identical, such as *Google Inc.* and *Google*. However, the *entity-frequency* fails when the name variations are abbreviations, such as *VW (Volkswagen)* or *BoA (Bank of America)*.

It is important to note that the features are heavily dependent on the off-the-shelf Flair NER tagger and the Radboud Entity Linker, as well as the Wikipedia 2019 knowledge base. Thus, using a different setup of these systems could possibly change the results from the feature selection experiments. For example, the *KB-entity-frequency* feature is derived from the Radboud Entity Linker, together with the Wikipedia 2019 knowledge base. Using a knowledge base specifically designed for companies and organizations could render the *KB-entity-frequency* feature meaningful for the classifiers. In other words, the features that were derived from the off-the-shelf systems might have inherited too much noise due to the errors from the systems.

Another interesting observation is the predictive power of the positional baseline on the salient classification task. The positional baseline results in a 0.695 f1-score on the test set, only 0.072 points under the highest-performing classifier and 0.074 points below the highest-performing classifier on the SNEE task. When examining the precision and recall of the positional baseline, the precision is comparable to the ensemble models but has a significantly lower recall, which is expected due to not all salient named entities being present in the first sentence of the article. The power of the positional baseline is also reported in related work in salience detection [30, 38], although not in the same magnitude as in this study.

5.1.2 Sequence Labeling

When analyzing the single-task sequence labeling models' results on the sequence labeling task, the BERT-based models were shown to outperform the BiLSTM-based models, both with and without the CRF layer. These results are in line with the original BERT paper [28] but is in contrary to the results derived in [33], where the authors approached the keyword extraction task as a sequence labeling task and found that using the output of BERT as input embeddings to a BiLSTM outperformed the approach of finetuning the BERT model. The authors state that their results could be due to the small size of the training data, indicating that there was not enough data to properly finetune the large BERT model, which was not the case for this study.

As previously stated in the results section, all neural sequence labeling models outperformed the CRF baseline in terms of f1-score. The precision of the CRF baseline is comparable to the evaluated sequence labeling models. However, the recall is significantly lower. This implies that the neural models are able to, in general, detect more salient named entities per document. One reason the neural sequence labeling models outperform the baseline CRF in terms of recall could be attributed to the models' ability to identify name variations of a salient named entity. However, a deeper analysis would need to be performed to confirm this hypothesis.

5.1.2.1 Multi-task Learning

The model that achieved the highest f1-score among the sequence labeling models had a multi-task learning setup. However, the increase in performance when using a multi-task learning setup was not consistent across all model architectures. The performance increased in the BERT-finetuned architecture but decreased in the BiLSTM-based architecture, making it difficult to draw any robust conclusions if the NER task is an appropriate auxiliary task for SNEE.

The LSTM-based models seem to fail to find a good representation that both captures the NER and SNEE task, without sacrificing the representation of the SNEE task. In multi-task learning, there is a risk that the model fails at specializing on the target task, due to the auxiliary task contaminating the learned features. One way of avoiding this situation is to use a soft parameter sharing architecture, where each task has its own task specific representation (as illustrated in figure 2.6). Experimenting with different multi-task learning model architectures, such as a soft parameter sharing or a hierarchical setup could allow the LSTM-based

models reach a performance gain. Another possible reason the multi-task learning setup decreased the performance on the SNEE task in the LSTM-based models is that the models favoured the NER task over the SNEE task. Due to each news article in the training set generally containing a higher amount of NER labels compared to SNEE labels, the models gain more when focusing on reducing the NER loss. Decreasing the weight (λ_t in equation 2.9) for the NER task's loss could hinder the models from favouring the NER task.

There are several other possible reasons why the multi-task learning setup did not show significant improvements across all model architectures. A possibility is that the NER task is simply not appropriate to use as an auxiliary task in SNEE. As stated in the method section, the NER task was chosen as an auxiliary task due to its similarity to the main task, hence the potential knowledge correlation between the two tasks. The NER task could either be redundant in the sense that the potential information learned from the NER task does not provide any additional information to the SNEE task.

5.1.3 Comparison of Approaches

Even though the classical two-stage approach outperformed the sequence labeling models on the extraction task in terms of f1-score, the performance gain is minimal. When analyzing the precision/recall per document for the two best performing models, namely the *two-stage random forest* and the *multi-task BERT-finetuned + CRF* model, it was observed that the two models' recall and precision per document were highly correlated, having a correlation coefficient of 0.65 for precision and 0.69 for recall. This indicates that the models generally fail in similar situations and have similar performance on each article. If one model performed well on documents that the other model generally fails on, an ensemble method could have been interesting to explore.

5.1.3.1 Error Analysis

In the following section, two news article examples are presented together with the corresponding predictions from a selection of the evaluated models, with the objective to highlight their behaviour. The body text of the first news article example is shown in figure 5.1. The ground truth salient named entities are marked in bold, namely: *Huawei*, *Nokia* and *Ericsson*. Table 5.1 presents the predictions from the best performing approaches, as well as the predictions from their corresponding baseline models. The table shows that the multi-task sequence labeling model was the only model to identify all ground truth salient named entities.

Table 5.2 presents a closer look at the two-stage approach's inner workings by displaying the classifier's SHAP values for *entity-frequency (EF)* and *first-sentence (FS)* as well as the predictions for each identified named entity in the news article example. As the table shows, *Huawei* was mentioned in the first sentence and had a total number of nine mentions in the news article, which resulted in a SHAP value of 0.16 and 0.21, respectively. Adding 0.16 and 0.21 to the base value of 0.5 resulted in a 0.87 probability that *Huawei* was salient. However, *Nokia* and *Ericsson* were both first mentioned in the 10th sentence, which significantly lowered the probability that they were salient (0.12%).

Figure 5.2 presents the second example of a news article that contain characteristics that all models, regardless of approach, often struggle with. This is because the models seem to have generally learned that all named entities found in the first sentence are salient, regardless of how many times the named entity is mentioned. This leads to all models producing the false positives: *Tesla* and *BMW*. These types of false positives occur fairly frequently and contribute to the overall error of the models. Preventing the two-stage approach from producing these errors would involve researching alternative features to introduce in the classification model. Features that describe what type of company/organization a named entity is, such as which industry it operates in, could be interesting to evaluate. For the neural sequence labeling models, it is less clear what could be done to prevent the previously mentioned false positives.

Perhaps experimenting with other entity-focused embeddings, such as Wikipedia2Vec[1] could produce different results.

> The Chinese embassy in Paris on Sunday urged the French government not to discriminate against **Huawei** as it selects suppliers for its 5G mobile network, saying it feared the company would face more constraints than rivals. China's **Huawei**, a global giant in telecoms network equipment, is the centre of an international political storm as the United States seeks to convince countries to ban the company from their mobile networks. Washington says its technology could allow "back doors" for Chinese spying - an allegation denied by **Huawei** and Beijing. France is in the early stages of rolling out its next-generation wireless technology, and the government's stance over **Huawei**'s possible role still lacks clarity, according to some telecoms industry trade bodies...
> ... The embassy also said that China had used foreign companies such as Finland's **Nokia** and Sweden's **Ericsson** to equip its own domestic networks. "We do not wish to see the development of European companies in China affected due to discrimination against **Huawei** and protectionism in France and other European countries," it said. France's cybersecurity agency ANSSI, which is scrutinising equipment from various suppliers, is due to issue its preliminary findings later this month. Some telecoms operators have already picked 5G equipment makers, with France's Orange opting for **Nokia** and **Ericsson**. Britain has granted **Huawei** a limited role in its 5G roll-out, while the European Union has resisted pressure from Washington for an outright ban in its guidance to member states. The United States suggested in recent days it could consider taking a stake in **Nokia** and **Ericsson** to counter **Huawei**'s dominance in 5G technology.

Figure 5.1: News article example 1. The ground truth salient named entities are highlighted in bold.

Approach	Type	Model	Extracted Named Entities
Two-stage	-	Random Forest	Huawei
Sequence labeling	Multi-task	BERT-finetuned + CRF	Huawei, Ericsson, Nokia
Two-stage	-	Positional baseline	Huawei
Sequence labeling	Single-task	CRF-baseline	Huawei

Table 5.1: Predictions on *News Article Example 1* from the best performing models in the two-stage approach and sequence labeling approach, together with their corresponding baselines' predictions.

NE	Salient	EF	FM	EF SHAP value	FM SHAP value	P(Salient)	Prediction
Huawei	True	9	1	+0.16	+0.21	0.87	True
Nokia	True	3	10	+0.08	-0.46	0.12	False
Ericsson	True	3	10	+0.08	-0.46	0.12	False
ANSSI	False	1	12	-0.14	-0.36	0.00	False
Orange	False	3	14	+0.13	-0.46	0.17	False

Table 5.2: SHAP values (for the features *entity-frequency (EF)* and *first-mention (FM)*) and predictions from the Random Forest classifier for each named entity in the example news article. The base value for the probability of a named entity being *salient* is 0.50.

The results show that the SNEE task can be approached in an end-to-end manner and modeled as a sequence labeling task, achieving comparable performance to the two-stage

[1] https://wikipedia2vec.github.io/wikipedia2vec/

> China's top lithium producer **Ganfeng Lithium**, a supplier to carmakers such as Tesla and BMW has priced its Hong Kong listing at the bottom of its marketed range, raising $421 million, according to a source involved in the deal. **Ganfeng** priced its offering at HK$16.50 ($2.11) per share, the bottom end of an indicative range of between HK$16.50 and HK$26.50, the source said.

Figure 5.2: News article example 2. The ground truth salient named entities are highlighted in bold.

Approach	Type	Model	Extracted Named Entities
Two-stage	-	Random Forest	Ganfeng Lithium, Tesla, BMW, Ganfeng
Sequence labeling	Multi-task	BERT-finetuned + CRF	Ganfeng Lithium, Tesla, BMW, Ganfeng
Two-stage	-	Positional baseline	Ganfeng Lithium, Tesla, BMW
Sequence labeling	Single-task	CRF-baseline	Ganfeng Lithium, Tesla, BMW, Ganfeng

Table 5.3: Predictions on *News Article Example 2* produced by the best performing models in the two-stage approach and sequence labeling approach, together with their corresponding baselines' predictions.

approach. This means that a neural sequence labeling model can effectively replace a NER tagger, NEL system and a classifier, which will be further discussed in the conclusion chapter. Another benefit of the sequence labeling approach is that it does not rely on engineered features, which can be costly to create. As illustrated in table 5.1, the sequence labeling approach is able to identify that *Ericsson* and *Nokia* are salient named entities, even though they are mentioned late in the article and have a reasonably low entity frequency. A reason why the sequence labeling model was able to achieve this could be attributed to the model learning the relationship between *Huawei*, *Nokia* and *Ericsson* and that they are often jointly salient. This can indicate a more robust model that can handle outliers better than the two-stage approach, which is bounded to the simple text features. However, incorporating a priori information can also be a disadvantage and lead to false positives.

5.2 Method

In the following section, the chosen methodology is discussed, with a focus on its potential weaknesses and limitations. Moreover, suggestions on improvements are presented. Lastly, the study's sources are discussed.

5.2.1 Data

The dataset used in this study was automatically created using the methodology described in section 3.2. Although related work in salience detection uses a similar approach when creating the training dataset, there are several potential drawbacks employing such method. The dataset creation approach used two off-the-shelf sequence labeling systems to automatically create the training data. These systems are prone to introduce errors in the data creation phase, either from the NER tagger mislabeling a named entity, or from the NEL system either failing to find the unique identifier for a named entity or failing to assign the correct unique identifier. For example, the Flair NER-tagger has a recall of 0.920, meaning that the NER tagger will, on average, fail to recognize roughly one out of ten named entities in the text. It is well known that supervised machine learning is heavily dependent on the quality of the data, which is also the case in this study. Therefore, using imperfect training data can negatively impact the models' performance, due to introducing too much noise in the data.

The hypothesis that the salient named entities in a news article can be found in the news articles' headline seems to hold when inspecting the precision of the taken training set gen-

eration approach on the test set, presented in table 3.3. The training set generation approach yields a precision of 0.918, meaning that the named entities found in the headline often are in fact salient named entities. However, the recall is 0.572, meaning that the named entities found in the headline of an article, on average, only cover roughly 57% of the ground truth salient named entities. The low recall suggests that taking an approach similar to [30], where the authors used news article abstracts, which generally contains more information than the headlines, would presumably increase the recall of the dataset generation approach. However, it may also decrease the precision. Whether to use the abstracts or the headlines when automatically creating a training set for SNEE ultimately depends on how strict one defines salience. Unfortunately, there were no abstracts available for the news articles used in this study. Therefore, the potential effects of using the abstracts in the dataset creation approach could not be explored.

Another possible weakness of the study is in the methodology taken when creating the human-annotated validation and test set. Only one human annotator performed the manual annotation. Having only one annotator can introduce bias in the resulting ground truth annotations and lead to deceptive model evaluation results. Taking an approach similar to [38], where the authors collected three different judgments per entity, could help in further substantiating the results and increasing the reliability and validity of the study.

As previously mentioned in section 3.1, defining salience can be difficult, due to it being subjective to the reader and often hard to define on a binary scale. The ambiguousness also increases when dealing with companies and organizations that have hierarchical structures. Consider the following news article example:

> *Company 1*, which is owned by *Company 2*, has been fined by the Securities and Exchange Commission for repeated misstatements that failed to disclose the firm's receipt of payments from their customers. According to the SEC's order, between 2015 and late 2018, *Company 1* made misleading statements and omissions in customer communications.

In this example, *Company 1* is clearly salient to the news. However, it is more challenging to define the salience of *Company 2*. *Company 2* is not the company being fined and therefore not the main focus of the article. However, it still plays some relevance in the text, because it is highly related to the company that is being sued. The author that decided to include the information about the relationship must have thought the information played some degree of importance in the context. In situations similar to the example above, it would be beneficial to have a definition of salience that is scaled, such as in [38].

Further exploring the ambiguousness of the task could provide more context to the results, for example by letting different human annotators annotate the test set and then calculate their precision/recall/f1-score on the ground truth annotations. Perhaps the evaluated models' performance is on par with the human annotators, which would imply that there is no more improvements to be made to the models' performance in terms of accuracy.

5.2.2 Model Development

The decision to use off-the-shelf systems for NER and NEL in the two-stage approach was taken due to the focus of the study being on feature selection and the classification stage. However, it is important to note that the two-stage approach's performance could possibly be further increased by, for example, training a NER tagger on the financial news article domain instead of using the general-domain Flair NER tagger. However, as the error analysis presented in 5.1.3 suggests, the errors from the two-stage approaches' predictions were often due to the classifiers' inability to discriminate, and not the NER tagger's ability to identify named entities.

It is well known that training deep learning models is far more time-consuming than training shallow machine learning models. There are plenty of hyperparameter settings to

choose from and other decisions that have to be taken, such as determining the loss function, learning rate and layer dimensions. In this study, the approach of exploring multiple model architectures was favored over focusing on only one architecture variant. Eight different neural sequence labeling model architectures were trained and evaluated. Thus, the hyper-parameters and settings for each individual model variant were not thoroughly explored due to time constraints. This means that the models may not have showcased their full potential, and that they were trained on non-optimal settings. Perhaps by focusing on a smaller set of models, which would permit more time for finding the optimal training settings, would aid in deriving a deeper understanding of the models' behaviour. For example, performing a thorough grid search on the loss weights for the task-specific losses in the multi-task learning models could help in understanding how much emphasis should be put on the auxiliary NER task, which in turn could infer the usefulness of the NER task.

5.2.3 Sources

As mentioned in the method chapter, the research of salient entity extraction is still a small area, resulting in a limited amount of previous work, making it challenging to find a diverse set of studies to base this study on. This challenge was partially tackled by drawing inspiration from related areas in information retrieval, namely keyword extraction. Nevertheless, the papers that this study is based on are all peer-reviewed conference papers published between 2014 and 2020. Moreover, the majority of sources used in the theory chapter are either original papers of the theory presented, or peer-reviewed scientific papers with multiple citations.

5.3 The work in a wider context

Salient named entity extraction can be used in a range of different information retrieval tasks, such as document search, entity relatedness, event mining and document summarization. Moreover, as described in the introduction chapter, determining the salient entities in news articles is an important step in the process of mining company actions and involvements. However, it is of great importance to consider how the textual data is created and by whom. Disinformation can pose a big threat to financial markets. For example, online message boards are often exploited by people to plant disinformation and trade on the effect of said information. Therefore, much time and effort should be put into finding trustworthy news sources and data.

Moreover, Natural Language Processing has recently gained great traction from the financial domain. NLP is being deployed across the industry, ranging from audit companies using NLP to help automate their audit processes, to asset managers utilizing NLP to predict market movements. While the adoption of analyzing unstructured financial data using NLP is becoming the norm, the risk of companies and organizations maliciously exploiting this situation grows. For example, suppose a company knows that their quarterly report will be analyzed by others using sentiment analysis models. The company can deliberately use words that the sentiment model considers having a high sentiment value in order to "trick" the models. This is closely related to the existing problem of "greenwashing", where companies try to portray their actions as environment-friendly only for the sake of marketing and reaching new customers. This connects to the importance of using reliable and unbiased sources for the data that is operated on.

6 Conclusion

The following chapter revisits the research questions and presents their corresponding answers as well as a conclusion of the study. Moreover, possible directions and areas in future work are also discussed.

6.1 Research Questions

The following section lists each research question together with their corresponding answers.

1. **Which combination of hand-crafted textual features yield the highest performance when used by a classifier for classifying salient named entities?**

   Six different features were evaluated, where all features had been proposed in related work in salient entity discovery. The feature combination that produced the highest f1-score on salience classification was using *first-mention* and *entity-frequency*, where *first-mention* is the index of the sentence in which the first mention of the named entity appears and *entity-frequency* counts the occurrence of the head word of the named entity in the document. Adding the remaining features to the model did not contribute to any performance gains.

2. **How does the sequence labeling approach perform on the salient named entity extraction task compared to the two-stage approach?**

   Eight different neural sequence labeling models were evaluated using a single-task and multi-task learning setup. The models were compared against the two-stage approach on the salient named entity extraction task. The evaluation showed that the two-stage approach outperformed the neural sequence labeling models. The difference in f1-score between the best performing two-stage approach and the best performing single-task sequence labeling model was 0.034, constituting a relative increase of 4.4%. Furthermore, the corresponding difference for the multi-task sequence labeling model was 0.01, meaning that the two-stage approach outperformed the best performing multi-task learning model by 1.27%.

3. **How does the performance of the sequence labeling models change when a multi-task learning approach is taken and an auxiliary named entity recognition task is added to the models?**

All single-task sequence labeling models were altered to include an auxiliary named entity recognition task using a multi-task hard-parameter sharing architecture. The experiment results showed that the multi-task learning approach increased the f1-score on the salient named entity extraction task for the BERT-based models, showing a 3.1% increase, but decreased the f1-score for the BiLSTM-based models, where the largest decrease in f1-score amounted to -3.5% on the salient named entity extraction task. However, despite the increase in performance being low, the increase is consistent across all BERT-based model architecture variants, giving a strong indication that a multi-task learning setup increases the performance on the SNEE task when finetuning BERT. However, a more thorough evaluation, perhaps on a larger test set would be necessary to derive any substantial conclusions.

To summarize, this study has explored the task of extracting salient named entities from text documents in the domain of financial news articles, where the named entities are companies or organizations. Three separate approaches were explored. The first approach was drawn from previous work in salient entity discovery. The two remaining approaches inherit a sequence labeling methodology, drawing inspiration from recent development in keyword extraction and named entity recognition, addressing the salient named entity extraction task in an end-to-end fashion. The results showed that the sequence labeling approaches were slightly outperformed by the classical two-stage approach. However, the small difference in performance between the two-stage approach and the best performing sequence labeling approach indicate that the task of salient named entity extraction can be addressed in an end-to-end fashion using sequence labeling. Using an end-to-end approach can effectively eliminate the need of external knowledge bases and computationally heavy NEL systems, hence eliminating the time-consuming feature generation stage. Moreover, modeling the task of salient named entity extraction as a sequence labeling task enables the possibility for it to be used as an auxiliary task in other multi-task learning setups. For example, the salient named entity extraction task could potentially be helpful when performing event extraction or document summarization.

This study has also shown that a suitable training set for the salient named entity extraction task can be automatically created using off-the-shelf NER and NEL systems and considering the named entities found in a news article's headline as salient. This creates a training set with strict salient named entity annotations, which has a high precision (0.944) but lower recall (0.572) on the manually annotated salient entity annotations.

6.2 Future Work

This study shows that the classical two-stage approach slightly outperforms the sequence labeling approach on the salient named entity extraction task. However, due to the small size of the test set, further evaluation of the models on larger test sets would be necessary to substantiate any conclusive claims. Moreover, further examining the validity of the validation and test set's manual annotations would also be necessary to conclude the ambiguousness of the task. This can be done by allowing several experts annotate the data sets and calculating the inter-agreement across annotations. The manually annotated test set can then be publicly released to facilitate further research in the area and introduce a gold standard test set. Furthermore, exploring other methods to automatically create the training set, such as using abstracts instead of headlines in the salience label creation phase could perhaps result in a corpus with a less strict notion of salience, which in turn would decrease the difference between the characteristics of the test and training set.

The varied performance gain observed when using a multi-task learning setup would be interesting to explore further. Experimenting with different multi-task learning architectures, as well as performing a thorough grid search over the model's hyperparameters could provide meaningful results. Moreover, it would also be interesting to experiment with adding

or using other auxiliary tasks that share similarities with the salient named entity extraction task, such as coreference resolution or relationship extraction. The coreference resolution task could possibly enable the model to better find name variations of named entities, while the relationship extraction task can promote attention on relationships that often involve salient named entities.

Bibliography

[1] PAUL C. TETLOCK. "Giving Content to Investor Sentiment: The Role of Media in the Stock Market". In: *The Journal of Finance* 62.3 (2007), pp. 1139–1168. DOI: https://doi.org/10.1111/j.1540-6261.2007.01232.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2007.01232.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.2007.01232.x.

[2] Alireza Mansouri, Lilly Affendey, and Ali Mamat. "Named Entity Recognition Approaches". In: *Int J Comp Sci Netw Sec* 8 (Jan. 2008).

[3] Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. "SEL: A Unified Algorithm for Entity Linking and Saliency Detection". In: *Proceedings of the 2016 ACM Symposium on Document Engineering*. DocEng '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 85–94. ISBN: 9781450344388. DOI: 10.1145/2960811.2960819. URL: https://doi-org.e.bibl.liu.se/10.1145/2960811.2960819.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[5] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].

[6] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6442–6454. DOI: 10.18653/v1/2020.emnlp-main.523. URL: https://www.aclweb.org/anthology/2020.emnlp-main.523.

[7] Archana Goyal, Vishal Gupta, and Manish Kumar. "Recent Named Entity Recognition and Classification techniques: A systematic review". In: *Computer Science Review* 29 (2018), pp. 21–43. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2018.06.001. URL: http://www.sciencedirect.com/science/article/pii/S1574013717302782.

[8] W. Shen, J. Wang, and J. Han. "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions". In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2015), pp. 443–460. DOI: 10.1109/TKDE.2014.2327028.

[9] L. A. Ramshaw and M. P. Marcus. "Text Chunking Using Transformation-Based Learning". In: *Natural Language Processing Using Very Large Corpora*. Ed. by Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky. Dordrecht: Springer Netherlands, 1999, pp. 157–176. ISBN: 978-94-017-2390-9. DOI: 10.1007/978-94-017-2390-9_10. URL: https://doi.org/10.1007/978-94-017-2390-9_10.

[10] Anthony Myles, Robert Feudale, Yang Liu, Nathaniel Woody, and Steven Brown. "An Introduction to Decision Tree Modeling". In: *Journal of Chemometrics* 18 (June 2004), pp. 275–285. DOI: 10.1002/cem.873.

[11] Vladimir Svetnik, Andy Liaw, Christopher Tong, John Culberson, Robert Sheridan, and Bradley Feuston. "Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling". In: *Journal of chemical information and computer sciences* 43 (Nov. 2003), pp. 1947–58. DOI: 10.1021/ci034160g.

[12] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. ISSN: 00905364. URL: http://www.jstor.org/stable/2699986.

[13] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777. ISBN: 9781510860964.

[14] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[16] R. J. Williams and D. Zipser. "Gradient-based learning algorithms for recurrent networks and their computational complexity". In: *Backpropagation: Theory, Architectures and Applications*. Ed. by Y. Chauvin and D. E. Rumelhart. Hillsdale, NJ: Erlbaum, 1992.

[17] Y. Bengio, P. Frasconi, and P. Simard. "The problem of learning long-term dependencies in recurrent networks". In: *IEEE International Conference on Neural Networks*. 1993, 1183–1188 vol.3. DOI: 10.1109/ICNN.1993.298725.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[19] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition". In: *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*. Ed. by Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 799–804. ISBN: 978-3-540-28756-8.

[20] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074. DOI: 10.18653/v1/P16-1101. URL: https://www.aclweb.org/anthology/P16-1101.

[21] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1558607781.

[22] Joseph Worsham and Jugal Kalita. "Multi-task learning for natural language processing in the 2020s: Where are we going?" In: *Pattern Recognition Letters* 136 (Aug. 2020), pp. 120–126. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2020.05.031. URL: http://dx.doi.org/10.1016/j.patrec.2020.05.031.

[23] Joachim Bingel and Anders Søgaard. "Identifying beneficial task relations for multi-task learning in deep neural networks". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 164–169. URL: https://www.aclweb.org/anthology/E17-2026.

[24] Soravit Changpinyo, Hexiang Hu, and Fei Sha. "Multi-Task Learning for Sequence Tagging: An Empirical Study". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2965–2977. URL: https://www.aclweb.org/anthology/C18-1251.

[25] Victor Sanh, Thomas Wolf, and Sebastian Ruder. "A Hierarchical Multi-Task Approach for Learning Embeddings from Semantic Tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 6949–6956. DOI: 10.1609/aaai.v33i01.33016949.

[26] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. "A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1923–1933. DOI: 10.18653/v1/D17-1206. URL: https://www.aclweb.org/anthology/D17-1206.

[27] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *ICLR*. 2013.

[28] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT*. 2019.

[29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144.

[30] Jesse Dunietz and Daniel Gillick. "A New Entity Salience Task with Millions of Training Examples". In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Gothenburg, Sweden: Association for Computational Linguistics, Apr. 2014, pp. 205–209. DOI: 10.3115/v1/E14-4040. URL: https://www.aclweb.org/anthology/E14-4040.

[31] Michael Gamon, Tae Yano, Xinying Song, Johnson Apacible, and Patrick Pantel. "Identifying Salient Entities in Web Pages". In: *Proceedings of the 22nd ACM International Conference on Information amp; Knowledge Management*. CIKM '13. San Francisco, California, USA: Association for Computing Machinery, 2013, pp. 2375–2380. ISBN: 9781450322638. DOI: 10.1145/2505515.2505602. URL: https://doi.org/10.1145/2505515.2505602.

[32] Chuan Wu, Evangelos Kanoulas, and Maarten de Rijke. "It all starts with entities: A Salient entity topic model". In: *Natural Language Engineering* 26.5 (2020), pp. 531–549. DOI: 10.1017/S1351324919000585.

[33] Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. "Keyphrase Extraction as Sequence Labeling Using Contextualized Embeddings". In: *Advances in Information Retrieval*. Ed. by Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins. Cham: Springer International Publishing, 2020, pp. 328–335. ISBN: 978-3-030-45442-5.

[34] Rabah Alzaidy, Cornelia Caragea, and C. Lee Giles. "Bi-LSTM-CRF Sequence Labeling for Keyphrase Extraction from Scholarly Documents". In: *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 2551–2557. ISBN: 9781450366748. DOI: 10.1145/3308558.3313642. URL: https://doi.org/10.1145/3308558.3313642.

[35] Matej Martinc, Blaž Škrlj, and Senja Pollak. "TNT-KID: Transformer-based neural tagger for keyword identification". In: *Natural Language Engineering* (2021), pp. 1–40. DOI: 10.1017/S1351324921000127.

[36] Evan Sandhaus. *The New York Times Annotated Corpus*. Version V1. 2008. DOI: 11272.1/AB2/GZC6PL. URL: https://hdl.handle.net/11272.1/AB2/GZC6PL.

[37] Chuan Wu, Evangelos Kanoulas, Maarten de Rijke, and Wei Lu. "WN-Salience: A Corpus of News Articles with Entity Salience Annotations". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 2095–2102. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.257.

[38] Milan Dojchinovski, D. Reddy, T. Kliegr, T. Vitvar, and H. Sack. "Crowdsourced Corpus with Entity Salience Annotations". In: *LREC*. 2016.

[39] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. "REL: An Entity Linker Standing on the Shoulders of Giants". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, 2020, pp. 2197–2200. ISBN: 9781450380164. DOI: 10.1145/3397271.3401416. URL: https://doi.org/10.1145/3397271.3401416.

[40] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139.

[41] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. DOI: 10.1017/CBO9780511802256.