Master thesis : Extending Joint Entity and Event Coreference Resolution across Docume

Auteur : Nelissen, Louis
Promoteur(s) : Ittoo, Ashwin
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en science des données, à finalité spécialisée
Année académique : 2021-2022
URI/URL : http://hdl.handle.net/2268.2/16441

University of Liège
Faculty of Applied Sciences

Academic year 2021-2022

Extending Joint Entity and Event Coreference Resolution across Documents

Master thesis presented in partial fulfillment of the requirements for the degree of Master of Science in Data Science and Engineering

Louis Nelissen

Supervisor: Prof. A. Ittoo

Extending Joint Entity and Event Coreference Resolution across Documents

Louis Nelissen

Supervisor: Prof. A. Ittoo

Abstract

Detecting corefering events and entities in texts is an important task in NLP, where it plays a role in many other tasks and applications. In this work, we build on a joint approach of entity and event coreference resolution, pioneered by H. Lee, Recasens, et al. 2012 and matured by Barhom et al. 2019 using a neural architecture. In particular we look at coreference resolution across documents, more complicated and less researched than coreference resolution within documents. Using Barhom et al. 2019's model, we propose a series of extensions to improve its results. This is done by increasing the amount of information provided to the model, in particular the joint nature of the modelling and by improving entity and event representation with the use of document embedding. As a secondary problem, we investigate ways to improve the model's time performance through compressing the mention representations. Our results are compared with other works tackling the problem of cross document coreference resolution on the ECB+ dataset, the standard dataset for cross document entity and event coreference resolution.

Acknowledgements

I would like to thank my supervisor, Prof. A. Ittoo and Judicaël Poumay for their help in making this work come to life.

I would also like to thank my friends, my family and my colleagues for their support during this intense period of my life.

Contents

Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CD	Cross Document (Coreference Resolution)
DL	Deep Learning
ELMo	Embeddings from Language Models
GRU	Gated Recurrent Unit
LM	Language Model
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
PCA	Principal Component Analysis
RNN	Recurrent Neural Networks
SRL	Semantic Role Labelling
WD	Within Document (Coreference Resolution)

Chapter 1

Introduction

The detection of coreferring events and entities, defined as textual spans referring to the same entity or event, is an important task in NLP. It plays a role in many other tasks and applications. The particular subtask of entity recognition within documents has taken up a majority of the research interest in this field, with the other variants receiving less attention.

In 2012, Lee et al. proposed a model for coreference resolution which jointly modelled entity and events. This joint modelling lead to an improvement in performance for both tasks. Barhom et al. 2019 used this joint modelling idea to propose a neural architecture for cross-document coreference resolution.

In this work, we build upon their model by adding a series of modifications in order to first improve its results on the ECB+ dataset, and second its time performance. These improvements center around 3 aspects. We accentuate the joint nature of the modelling by increasing the frequency of switches between entity and event coreference training iterations. We also broaden entity and event representation with the use of document embedding. Finally we investigate the compression of mention representations with the aim to improve time performances.

In chapter 2 we will discuss background concepts. First, we will go over the concepts of Natural Language Processing and in particular the task of coreference resolution. This will be followed by a brief explanation of different tools and models used in this work. Chapter 3 will cover recent works tackling coreference resolution. Chapter 4 will detail our proposed extensions, chapter 5 the implementation, data and training procedures used and chapter 6 the performance of the aforementioned solution and further lines of inquiry. Chapter 7 will conclude this paper. Additional tables can be found in the Appendix.

Chapter 2

Background

This chapter will go over important theoretical and practical notions supporting the solution we propose. We first introduce Natural Language Processing and coreference. We then focus on the task of coreference resolution in its different variants.
We then go over two important categories of models we will use in this work: word embedding and document embedding. To explain how the models work, we will present the Recurrent Neural Network (RNN), Encoder-Decoder and Transformer architectures.
Finally we introduce the concept of dimension reduction and two approaches we will use (PCA and deterministic bottlenecked autoencoders).

2.1 Theoretical Notions

2.1.1 Natural Language Processing

Natural Language Processing (NLP), also called Computational Linguistics, is one of the possible applications of Artificial Intelligence (AI). It focuses on the treatment of natural language[1] by computers to perform useful tasks (in a similar fashion computer vision centres on the treatment of visual input by computers) (Deng and Y. Liu 2018). NLP is a multidisciplinary field, which, on top of AI, also involves elements from fields such as linguistics and computer science (Eisenstein 2019). The complexity of this field resides in the ambiguity and variability of human language (Goldberg 2017).

The development of NLP can historically be divided in 3 periods: rationalization, empiricism and deep learning.
The *rationalism* period coincides with the early phase of AI development in the 1950s, relying on the notion that there must be an innate structure to language (Chomsky 2015). This period is also referred to as the symbolic period as

[1]Natural language refers to a language that has developed as a means of communication between people, rather than a language created for computers for instance (*Natural Language* 2021)

programs were built using symbolic logic rules. The programs would then apply these rules to a given input and produce an output. The main drawback of these techniques is their inability to learn from data.

This lead to the development of the *empiricist* NLP models around the 1980s, coinciding with the rise of machine learning. Instead of trying to use the innate structure of natural languages, the empiricist approach exploits data to search for patterns and generalizations. Building on theoretical concepts such as the Hidden Markov Model (Baum and Petrie 1966), models like support vector machines, conditional random fields and perceptrons became the norm. These models are usually referred to as statistical models.

Since the 2010s, these models are being supplanted by the so-called *"deep" learning models* (Y. Bengio 2009), (LeCun, Yoshua Bengio, and G. Hinton 2015), (Goodfellow, Yoshua Bengio, and Courville 2016). These models are characterized by their ability to process huge amounts of data. They also do not need a lot of the feature engineering[2] that was required by older models (Deng and Y. Liu 2018). On the other hand, they require a significant amount of data to train.

Research in NLP is usually focused on certain tasks. The most common of these include: speech recognition, spoken language understanding, dialogue systems, lexical analysis, parsing, machine translation, information retrieval, question answering, sentiment analysis, social computing, natural language generation, and natural language summarization (Deng and Y. Liu 2018). The particular task of coreference resolution and its two main categories of entity coreference resolution and event coreference resolution (detailed in Sections 2.1.2 and 2.1.2 respectively) are of importance as they serve as building blocks for several other tasks and applications. Information extraction, text summarization, machine translation and text mining are some examples that sometimes make use of coreference resolution (Poumay and Ittoo 2021).

2.1.2 Coreference

Coreference is a linguistic concept that occurs when two or more expressions refer to the same entity(Group 2022). These two expressions are then referred to as being coreferential. For example, in the sentence 'John burnt himself while cooking', 'John' and 'himself' are coreferential as they both refer to the same entity, in this case a person named John.

Entities in natural language processing systems (and humans) are interpreted with respect to a discourse model (Karttunen 1969). A discourse model is a mental model built incrementally by the understander of some text, which contains representations of the entities as well as the relationships between them. An entity is introduced in this model the first time it is mentioned, a process known as evocation. When the text subsequently refers to this entity,

[2]Feature engineering refers to the concept of leveraging domain knowledge when crafting Machine Learning features from raw data (Zabokrtsky 2016).

the understander links this mention of the entity to the entity in the discourse model. This process is illustrated in Figure 2.1.

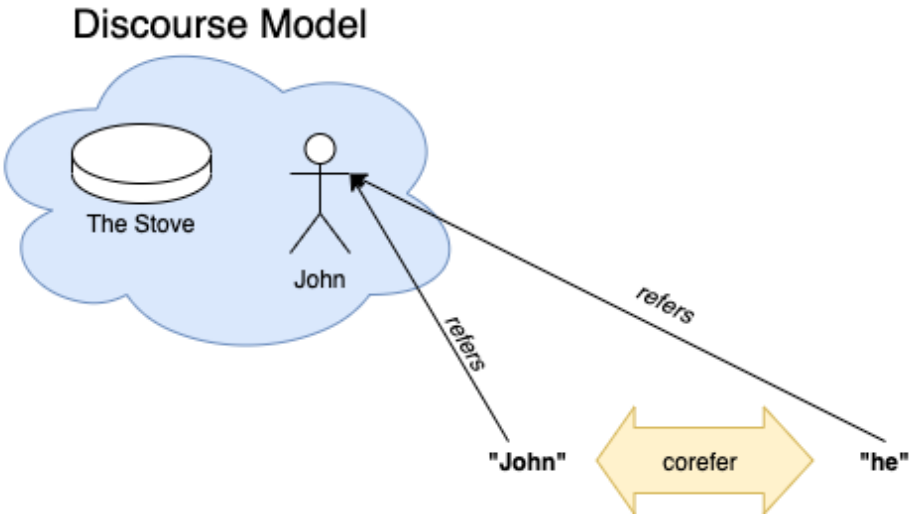

Figure 2.1: Discourse model example for the sentence ”John burnt himself while cooking”

As mentioned previously, coreference is an important component of Natural Language Processing, and is essential to several important tasks. For example, a question answering system that needs to process an encyclopedia entry about J.R.R Tolkien must understand that, in the sentence ”He wrote the Lord of the Rings.”, ”he” refers to J.R.R. Tolkien to answer any question concerning the author’s bibliography properly.

Another example is in machine translation, when translating to English from a language which drops pronouns such as Italian or Spanish. To translate the Italian sentences: ”La professoressa ha chiamato. Dice che sta arrivando.”[3], the machine translation system must correctly understand that ”dice” refers to the female professor mentioned in the first and translate it by ”she says”. Incorrect coreference might lead to improper translation of ”dice” into ”he says”. This is a problem observed in some machine translation systems (Lopez Medel 2021).

Entity coreference

Entity coreference resolution refers to the problem of finding and clustering together all expressions that refer to the same entity in the discourse model. In the following two sentences: ”J.R.R. Tolkien is an author. He wrote the Lord of the Rings.”, entity coreference resolution consits in linking the two mentions refering to the discourse entity of J.R.R. Tolkien: ”J.R.R. Tolkien” and ”he. Despite being well studied, it is considered one of the more difficult tasks of natural language processing (Stylianou and Vlahavas 2021).

[3]”The (female) professor called. She says that she is arriving.”

Event coreference

Event coreference resolution is another variant of coreference resolution. Instead of clustering entities referring to the same discourse entity, the task is to cluster together event mentions referring to the same event. The definition of an event is not strictly settled and is defined differently depending on the dataset[4] Most relevant is the definition of events proposed by Cybulska and Vossen 2014 in the ECB+ dataset models used in this paper, which defines events as a combination of four components:

1. an action component which describes **what** happens
2. a time component which anchors the action in time. It describes **when** the action happens
3. a location component which anchors the action in space. It describes **where** the action happens
4. a participant component which links the event to an entity or entities which is/are involved, undergo(es) a change as a result of or facilitate(s) an event or a state.

The following two sentences refer to the same event, namely the writing of J.R.R. Tolkien's The Lord of the Rings:

1. The Lord of the Rings was **written** while Tolkien was a professor at Oxford.
2. The **redaction** of the Lord of the Rings was a lengthy process.

where **written** and **redaction** are the mentions of this event.

Resolving event coreferences is considered more complex than entity coreferences, as events mentions can also be verbal instead of just nominal as with entities (Jurafsky and Martin 2009).

Mention Detection

A related task to coreference resolution is the mention detection task. It is an essential preprocessing step for coreference resolution, as it centres around the identification of mentions of entities in text (J. Yu, Bohnet, and Poesio 2020). It is often tackled as a separate task to coreference resolution. Some recent models have started to tackle both tasks together in the case of entity coreference resolution (K. Lee, He, Lewis, et al. 2017; Joshi, Levy, et al. 2019) and event coreference resolution (Y. Lu et al. 2022). This is important to note when comparing resulting and evaluating these models, as coreference

[4]The Automatic Content Extraction program defines an event as "a specific occurrence of something that happens, often a change of state, involving participants" (*ACE (Automatic Content Extraction) English Annotation Guidelines for Events* 2005). The TimeML dataset describes events as "situations that happen or occur that can be punctual or durational, as well as stative predicates describing states or circumstances in which something obtains or holds true" (Pustejovsky et al. 2004).

resolution errors may be due to propagated mention detection errors (J. Lu and Ng 2020).

2.2 Practical Notions

2.2.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) refer to a family of artificial neural networks, known for their capacity to maintain a memory of past inputs. Based on the work of (Rumelhart, Geoffrey E. Hinton, and Williams 1986), RNNs are based on feed forward[5] neural networks, which are modified by adding a cycle. This allows RNNs to maintain an internal state (Elman 1990). This internal state, also called the recurrent state h_t, can serve as a memory, which is updated after every input. This property makes RNNs particularly suited to sequences of inputs of varying lengths, which is why they are one of the most used deep-learning tools in NLP (Fleuret 2021).

More formally, an RNN takes as input an ordered sequence of n d_{in}-dimensional vectors $x_{1:n} = x_1, x_2, \ldots, x_n, (x_i \in R^{d_{in}})$ and returns as output a single vector $y_n \in R^{d_{out}}$ of d_{out} dimension.

$$\begin{aligned} y_n &= \mathrm{RNN}\left(x_{1:n}\right) \\ \boldsymbol{x_i} \in \mathbb{R}^{d_{in}} \quad \boldsymbol{y_n} &\in \mathbb{R}^{d_{\mathrm{out}}} . \end{aligned} \tag{2.1}$$

This means we can implicitly define an output vector y_i for every sequence $x_{1:n}$. We can mathematically refer to the function returning these intermediary outputs as RNN*:

$$\begin{aligned} \boldsymbol{y}_{1:n} &= \mathrm{RNN}^{\star}\left(\boldsymbol{x}_{1:n}\right) \\ \boldsymbol{y_i} &= \mathrm{RNN}\left(\boldsymbol{x}_{1:i}\right) \\ \boldsymbol{x_i} \in \mathbb{R}^{d_{in}} \quad \boldsymbol{y_i} &\in \mathbb{R}^{d_{\mathrm{out}}} . \end{aligned} \tag{2.2}$$

We can also define an RNN recursively using a function R which takes as input a recurrent state h_{i-1} and an input vector x_i, and returns as output a new recurrent state h_i from which we can map an output vector y_i using a deterministic function:

[5]Meaning they do not form cycles

$$
\begin{aligned}
\mathrm{RNN}^{\star}\left(\boldsymbol{x}_{1:n}; \boldsymbol{h_0}\right) &= \boldsymbol{y}_{1:n} \\
\boldsymbol{y_i} &= O\left(h_{\boldsymbol{i}}\right) \\
\boldsymbol{h_i} &= R\left(\boldsymbol{h_{i-1}}, \boldsymbol{x_i}\right) \\
\boldsymbol{x_i} \in \mathbb{R}^{d_{\text{in}}}, \boldsymbol{y_i} &\in \mathbb{R}^{d_{\text{out}}}, \boldsymbol{h_i} \in \mathbb{R}^{f(d_{\text{out}})}.
\end{aligned}
\tag{2.3}
$$

Graphically RNNs can be represented "rolled" following the recursion or "unrolled" through time, detailing every time instant. This can be observed in Figure 2.2.

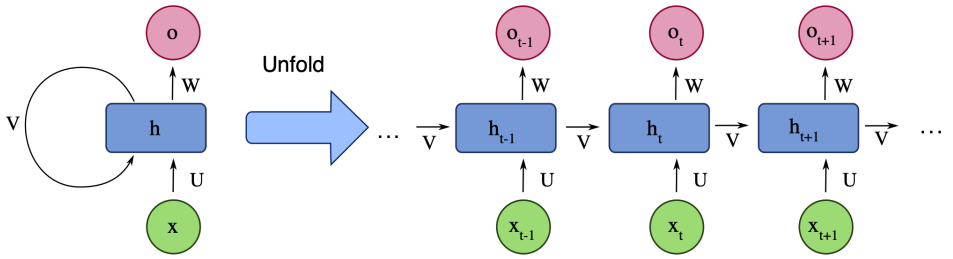

Figure 2.2: Rolled (recursive) and Unrolled (or unfolded) views of a Recurrent Neural Network. $x_i \in \boldsymbol{x}$ and $o_i \in \boldsymbol{o}$ are the inputs and outputs across time, $h_i \in \boldsymbol{h}$ the hidden states. V and W are the previously mentioned R and O functions. U is a function filtering the input. Courtesy of Deloche 2017b.

Nowadays two RNN architectures are used by the research community (Goldberg 2017). Both of these arose to tackle the *vanishing gradient problem*. This problem, shared by other deep network architectures, means that gradients[6] diminish quickly during backpropagation (Pascanu, Mikolov, and Yoshua Bengio 2013). Networks suffering from this problem are unable to capture long-range dependencies (Goldberg 2017).

The first commonly used RNN architecture is the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997), which introduces a gating system to tackle the vanishing gradient problem. LSTM augments its hidden state h_t with a memory cell c_m. The value of c_m, computed at time m, is the gated sum of the previous value c_{t-1} and z, an update computed from the current input x_m and the previous hidden state h_{t-1}. The next recurrent state h_m is computed from c_m. The memory cell is able to propagate information over long distances as its value is not passed through a non-linear squashing function (Eisenstein 2019). The original LSTM was mathematically defined in the following manner:

[6] A gradient is a derivative of a function with multiple input variables. Gradients are used in an optimization process known as gradient descent. Gradient descent is used to optimize neural neutworks.

$$
\begin{aligned}
\boldsymbol{h_t} = \boldsymbol{R}_{\mathbf{LSTM}}\left(\boldsymbol{h_{t-1}}, \boldsymbol{x}_t\right) &= \left[\boldsymbol{c_t}; \boldsymbol{h}_t\right] \\
\boldsymbol{c_t} &= \boldsymbol{f} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i} \odot \boldsymbol{z} \\
\boldsymbol{h_t} &= \boldsymbol{o} \odot \tanh\left(\boldsymbol{c_t}\right) \\
\boldsymbol{i} &= \sigma\left(\boldsymbol{x_t}\boldsymbol{W^{xi}} + \boldsymbol{h_{t-1}}\boldsymbol{W^{hi}}\right) \\
\boldsymbol{f} &= \sigma\left(\boldsymbol{x_t}\boldsymbol{W^{xf}} + \boldsymbol{h_{t-1}}\boldsymbol{W^{hf}}\right) \\
\boldsymbol{o} &= \sigma\left(\boldsymbol{x_t}\boldsymbol{W^{xO}} + \boldsymbol{h_{t-1}}\boldsymbol{W^{ho}}\right) \\
\boldsymbol{z} &= \tanh\left(\boldsymbol{x_t}\boldsymbol{W^{xz}} + \boldsymbol{h_{t-1}}\boldsymbol{W^{hz}}\right) \\
\boldsymbol{y_t} = O_{\text{LSTM}}\left(\boldsymbol{h_t}\right) &= \boldsymbol{h_t} \\
\boldsymbol{h_t} \in \mathbb{R}^{2 \cdot d_h}, \boldsymbol{x_i} \in \mathbb{R}^{d_x}, \boldsymbol{c_t}, \boldsymbol{h_t}, \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}, \boldsymbol{z} \in \mathbb{R}^{d_h}, \boldsymbol{W^{x\circ}} \in \mathbb{R}^{d_x \times d_h}, \boldsymbol{W^{h\circ}} \in \mathbb{R}^{d_h \times d_h}
\end{aligned}
\tag{2.4}
$$

$\boldsymbol{i}$, $\boldsymbol{f}$ and $\boldsymbol{o}$ are the gates mentionned previously, controlling for input, forget and output.

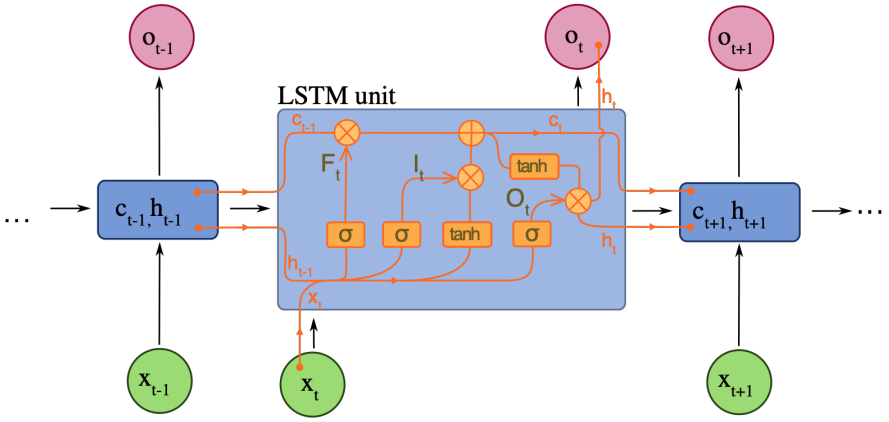

Figure 2.3: Representation of an LSTM cell. $x_i \in \boldsymbol{x}$ and $o_i \in \boldsymbol{o}$ are the inputs and outputs across time, $h_i \in \boldsymbol{h}$ the hidden states and $c_i \in \boldsymbol{c}$ the cell states. σ represents a sigmoïd gate and *tanh* a hyperbolic tangent gate, $\oplus$ element-wise addition gate and $\otimes$ element-wise multiplication gate. Courtesy of Deloche 2017a.

The other common architecture is called Gated Recurrent Unit (GRU) (Cho et al. 2014). Based on LSTM, GRU was designed to have fewer gates, making it computationally cheaper. This tackles one of the downsides of LSTM: its complexity makes it computationally expensive to work with.

2.2.2 Encoder-Decoder

Since the advent of DL, NMT systems have been implemented using two Recurrent Neural Networks. These two RNNs, when put end to end, form what is commonly known as an Encoder-Decoder (or sequence-to-sequence) architecture (Cho et al. 2014), (Sutskever, Vinyals, and Le 2014). In this

framework, the first RNN converts elements of the source language (a word, a sentence, ...) into a vector (or matrix) representation. The second RNN takes the vector representation as input and produces an output in the target language (Eisenstein 2019). The two RNNs are referred to as the encoder and decoder respectively. The other name used for this architecture, sequence-to-sequence (or seq2seq), refers to the fact that the strength of this architecture is its ability to operate on sequences of inputs and outputs of different lengths. This is complicated to do with a vanilla RNN architecture (Sutskever, Vinyals, and Le 2014). The architecture can mathematically be represented in the following manner:

$$\boldsymbol{z} = ENCODE(\boldsymbol{x}) \tag{2.5}$$

$$\boldsymbol{y}|\boldsymbol{x} \sim DECODE(\boldsymbol{z}), \tag{2.6}$$

Equation 2.6 means that the $DECODE(\mathbf{z})$ function defines the conditional probability $P(\mathbf{y}|\mathbf{x})$. In other words, for every source sentence, the encoder generates T_x hidden states:

$$h_i = f(h_{i-1}, x_i) \tag{2.7}$$

where $h(i)$ is the hidden state at time i, and f is a function representing the recurrent unit of the RNN. From this hidden state $h(i)$ the decoder generates an output sentence y in the target language.

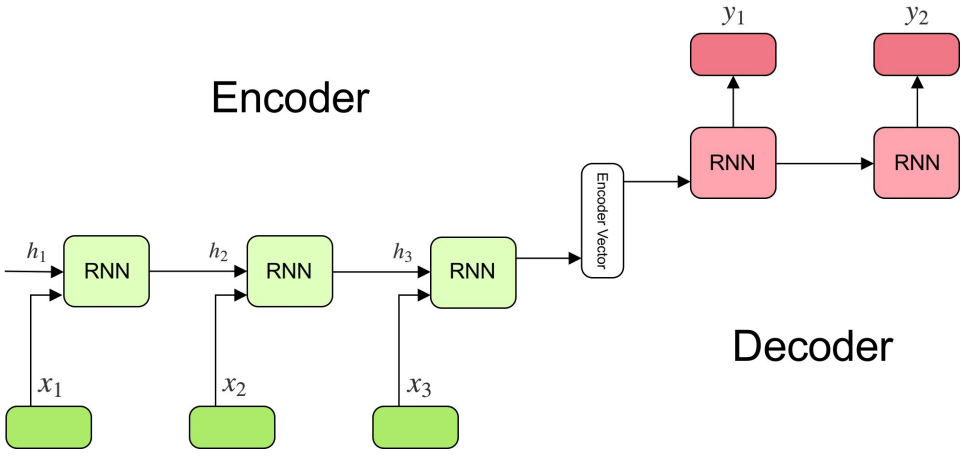

Figure 2.4: Encoder-decoder model (Kostadinov 2019)

2.2.3 Transformer

Vaswani et al. 2017 introduced a new framework called transformers to NLP. Transformers appeared as the evolution of the encoder-decoder architectures with RNNs that were state-of-the-art at the time, addressing several of its

weaknesses such as its difficulty with long-range dependencies and inability to be parallelized. Transforms on the other hand forego RNNs and rely solely on self-attention mechanisms.

Attention

Attention is a mechanism originally designed for Encoder-Decoder architectures. It allows the model to focus on parts of the input sequence which bring more value, especially with noisy inputs. Introduced by Bahdanau, Cho, and Yoshua Bengio 2016 to improve Neural Machine Translation, attention proved very successful as it was able to spread information throughout the entire sequence, allowing the decoder to selectively retrieve it. An attention model considers a certain window of words of size T_x. For each of these words, a context vector c_i is generated using as input all the words in the context. These context vectors will be given to the decoder as input. Using a series of attention weights α_{ij} and the hidden states generated by the encoder h_i, the context vectors are generated as follows:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.8}$$

These attention weights are computed using the normalized output of an MLP a given the hidden state of an RNN hidden state s_{i-1} just before emitting output y_i:

$$\begin{aligned} \alpha_{ij} &= \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}, \\ e_{ij} &= a\left(s_{i-1}, h_j\right) \end{aligned} \tag{2.9}$$

Other papers have proposed different types of attention, such as Luong, Pham, and C. D. Manning 2015.

Self Attention

Self attention is introduced by Vaswani et al. 2017 as part of their transformer model. They use an Encoder-Decoder setup with a novel architecture for both encoder and decoder. Unlike Bahdanau, Cho, and Yoshua Bengio 2016's attention which applies between an input sequence and an output sequence, self attention only applies to the input sequence or the output sequence on themselves. Given queries q, key vectors k and value vectors v derived from the input using three sets of weights W^Q, W^K and W^V and stacked in matrices Q,K and V respectively, self attention (also called Scaled Dot-Product Attention by the authors) is computed as follows:

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

Multi-head attention joins the information from different representations at different positions into a single input for an MLP to process. Vaswani et al. 2017 express it as follows:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_{\text{h}}\right) W^O \\ \text{where head}_{\,i} &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.11}$$

As can be observed in Figure 2.5, the encoder and decoder are made of layers (N of them). The encoder process embedded inputs through a multi-head self-attention sub-layer and an MLP. The decoder processes the outputs through a multi-head self-attention sub-layer, then combines it with the output of the encoder into another multi-head self attention and then into an MLP. As the Transformer is made of N parallel layers, it can easily be parallelized.

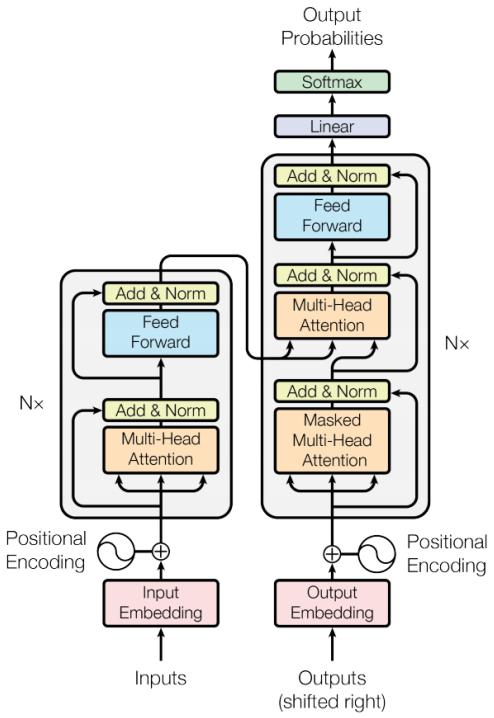

Figure 2.5: Transformer architecture (Vaswani et al. 2017)

2.2.4 Word Embedding

Representing natural language into inputs for an NLP model is no trivial task. The desired result is usually that the final representation is in the form of vectors of numbers so as to be readable by computer models. Additionally many machine learning algorithms require their inputs to be of fixed length, which is not a property of natural text (Le and Mikolov 2014). A straightforward approach like one-hot encoding is easy to implement but suffers both from the

curse of dimensionality[7] as well as an absence of semantic meaning (Ittoo 2019). In one-hot encoding, every word can be represented by a vector of the size of the vocabulary. Every word in the vocabulary has an assigned index, therefore a word is represented by a vector filled with 0s except for its corresponding index where the value is 1. The size of the vocabulary can lead to huge vectors (hence the curse of dimensionality) which do not carry any meaning: it is impossible to infer semantic relationships between words.

This thought process lead the field to reflect about the exact meaning and origin of semantics. One popular subfield of NLP, distributional semantics, predicates on the idea that a word's meaning is derived from its use. It relies on what is known as Firth's distributional hypothesis: that words used in similar contexts have similar meanings (Firth 1935; Harris 1954). As such, incorporating context into word representations should allow that representation to convey semantics (Kamath, J. Liu, and Whitaker 2019). There are broadly two ways of doing this: using count-based methods (such as using Pointwise Mutual Information or Latent Semantic Analysis) or neural-based methods.

The first to develop a successful neural-based method for encoding natural language was (Yoshua Bengio et al. 2003), who built models which reduced the dimensionality of word vectors by learning a distributed representation of words. These new representations are commonly known as embeddings.

word2vec

In 2013, Mikolov, K. Chen, et al. 2013 proposed 2 new neural techniques to compute continuous representations over large datasets. These new techniques were a lot more computationally efficient than previous techniques, and able to handle much larger corpora (in the ranges of billions of words) with much larger vocabularies (in the 10 000s to 100 000s). An added feature was that the new vectors possessed translational properties related to semantics. Applying simple arithmetic functions to these embeddings produced semantically relevant results. A classic example is the following: the word *queen* can be found from the embeddings of the words *king*, *man*, and *woman* by searching for the nearest vector to the vector sum:

$$v(queen) \approx v(king) - v(man) + v(woman) \tag{2.12}$$

The first technique, known as continuous bag of words (CBOW) and skip-gram, learns to predict a word given its context (i.e. the surrounding words). This leads to similar embeddings for words occurring in similar contexts. The second technique, named skip-gram, works the other way around: given a target word, the model predicts context words.

[7]The curse of dimensionality is a phenomenon observed when working with high-dimensional data (Bellman 1984). High dimensions often ends up being sparse, which is an undesirable property for a lot of learning models as it hinders them in finding patterns in increasingly distinct and unique data elements (Trunk 1979)

GLoVe

GLoVe is another word embedding technique published in 2014 by Stanford researchers (Pennington, Socher, and C. Manning 2014). Its fundamental difference with word2vec is its use of co-occurrence of words. The word2vec models learn embeddings by connecting target words to their context, but it does not take into account the frequency with which different context words appear. A frequent co-occurrence of terms in word2vec generates more training instances but contains no more information. GLoVe, on the other hand, emphasizes that the frequency of co-occurrences is important data. Instead of using them as extra training instances, the word embeddings of GLoVe connects a combination of word vectors directly to the likelihood that these words will occur together in the corpus.

FastText

FastText, proposed by Facebook AI Research in 2017 (Bojanowski et al. 2017), is an extension of word2vec. Its main novelty is to represent each word as an n-gram[8] of characters instead of learning vectors for words directly. For example, take the word, "business" with $n = 4$, the fastText representation of this word is $< busi, usin, sine, ines, ness >$. This has several advantages: it allows the embeddings to understand prefixes, suffixes and composed words, but also with rare or even unknown words. Despite not having learnt a specific word, it can infer an embedding and a meaning from its n-grams. This sets fastText apart from the other two commonly used word embedding methods (word2vec and GLoVe) which cannot process unknown words.

ELMo

A shared feature of word2vec, GLoVe and FastText is that they are all fixed embeddings: once trained, a given word will have a single word embedding even if the word has multiple meanings depending on its context. Peters et al. 2018 therefore introduce Embeddings from Language Model (ELMo), a Language Model based on bi-directional LSTM trained to predict words given its context. In other words, context is not only looked at during training but also during inference, which allows ELMo to provide a context-specific embedding for a given word. ELMo and those building upon it are therefore referred to as contextualized word embeddings.

BERT

Bidirectional Encoder Representations from Transformers (BERT), introduced by Devlin et al. 2019, is a powerful language model. It incorporates many of the innovations that were developed in NLP at the time. In practice, BERT

[8] In NLP, an n-gram is a contiguous sequence of n items from a text. These items can be words, letters or syllables depending on the case. A unigram is an n-gram with $n = 1$ (Ittoo 2019).

functions as a stack of Transformer-based encoders, trained to contextually embed words. It uses two different training strategies which are trained together. The goal of training is to minimize their combined loss. Its first training strategy is referred to as Msked Language Model, meaning that during training part of the word tokens supplied are replaced by a [MASK] token. Based on the context of the masked words, BERT attempts to predict the original values of the masked words. Using the output of this encoded outputs, an MLP classifier predicts the masked word. Unlike directional models like RNNs and LSTM, the Transformer at the heart of BERT allows it to read an entire sequence of words at once. It takes as training signal only the prediction of the masked value, meaning training takes longer as the model takes longer to converge. But this is offset by the model developing an increased awareness for the context of words. The second training strategy is called Next Sentence Prediction. In this training sequence, BERT learns to predict whether two sentences follow eachother in a text or not.

One of its most attractive features is that with some small modifications (refered to as fine-tuning), a pre-trained BERT can be optimized for a particular task while maintaing the majority of the model's weights intact. As BERT is a big model (345 million parameters), this allows for an easy access to a powerful model with relatively little modifications and extra training to a wide variety of NLP tasks.

2.2.5 Document Embedding

As mentioned previously, many ML models need their inputs to be fixed-length. Word embeddings translate words into fixed-length vectors, but in some cases we may want the model to take a series of words as a singular input. This has lead to the development of document embedding. By document we refer to any kind of series of two or more words. This includes sub-sentences, sentences and paragraphs. Document embeddings, like word embeddings, can provide richer inputs to ML models, and have been used in tasks such as text classification (Le and Mikolov 2014), document similarity tasks (Dai, Olah, and Le 2015) and paraphrase detection (Kiros et al. 2015). The field of document embedding has enjoyed a resurgence with the advent of deep learning around 2014, similarly to the field of word-embedding.

Word frequency-based document embedding

The classical way to encode documents was proposed in 1954 in Harris 1954. In this paper, Harris represents documents as "bag-of-words": the sum of the one-hot vectors of every unique word in the document (implying the vector contains either 0 values if the word is not present or 1 if the word is present at least once). This representation loses the sentence structure as well as the frequency of words[9].

[9]While one would instinctively sum the one-hot encoding vectors of ALL words to incorporate word frequency in the representation, this does not lead to a better representation

One extension of this idea is that of "bag-of-n-grams", a generalisation of bag-of-words. Wheras bag-of-words uses unigrams, bag-of-n-grams uses a vocabulary made of concatenations of n words (Ittoo 2019). This approach leverages the frequency of short word sequences and mitigates the loss of information due to the absence of sentence structure in the encoding without entirely removing it.

Unsupervised document embedding

Modern document embedding techniques are inspired by the improvements made in the field of word embedding by neural probabilistic language models, which are based on Firth's distributional hypothesis (see Section 2.2.4). The general idea behind these models is to extend this distributional hypothesis to documents and longer sequences of words

Mikolov, Sutskever, et al. 2013 were the first to apply this idea in trying to extend their skip-gram word2vec algorithm. With the observation that some phrases are more than the sum of their parts (for example it makes more sense to consider the phrase *Brussels Airlines* as a single token rather than two separately as it refers to a separate semantic concept), the authors incorporate a scoring method to decide when to consider words separately as phrases instead of unigrams [10]. Their results show that this allows for the processing of phrases 2 or 3 words long. Processing larger phrases would require too much computing power as the vocabulary size would be too large.

The next logical approach to embed documents or phrases, after considering phrases as single tokens, is to find ways to combine the word embeddings of the constituent words of phrases. The straightforward method is to simply average the word embeddings of the words, in a bag-of-words fashion. This method benefits from its simplicity and was a strong baseline for a long time, in tasks like answer sentence selection (L. Yu et al. 2014) or text similarity (Kenter and de Rijke 2015). It nonetheless suffers from several weaknesses, including the fact that its bag-of-word structure completely forgoes taking into account word ordering and other structural information that are inherent to documents (L. Yu et al. 2014). This straightforward method of averaging word embeddings also suffers from the fact that the word embeddings are not specifically optimized for representing sentences or documents. Several papers address this issue and propose models around this.

Kenter, Borisov, and de Rijke 2016 in their paper present a new model architecture, called Siamese CBOW, in which a series of word embeddings are averaged and trained simultaneously. The model therefore optimizes the word embeddings in the context of sentence embedding and not word embedding. Siamese CBOW outperforms the previously discussed baselines of averaging

of the document. Common words like "to", "the" or "a" have a high frequency while not having a high semantic importance (Pang, L. Lee, and Vaithyanathan 2002)

[10] The score is based on how often certain pairs of not too common words occur simultaneously

pre-trained word embeddings and (Mikolov, Sutskever, et al. 2013)'s skip-gram model.

Le and Mikolov 2014 introduce a different approach to the problem: they generalize word2vec's method to work with word sequences. The resulting models, christened Paragraph Vectors by the authors but commonly referred to as doc2vec, exist in 2 variants: Distributed Memory (PV-DM) and Distributed Bag of Words (PV-DBOW).

The Distributed Memory model is the pendent of continuous bag-of-words from word2vec: using an encoder-decoder architecture with an added memory vector, the model is trained to predict a word from its context. The memory vector, called the paragraph vector, is added at the beginning of every document and aims to capture the document's topic, or context from the input. Unlike word2vec, a word's doc2vec context only refers to the words preceding it and not surrounding it. Every word in the context is embedded to a word-embedding, then the sum of all these vectors is used as a feature for a classifier to predict the guessed word.

The Distributed Bag of Words model is the parallel to the skip-gram model from word2vec: the classification task is to predict a single context word using only the paragraph vector. In short, the context words are all dropped and the model is made to predict randomly sampled words in the document. PV-DBOW has been shown to train faster than PV-DM and require less memory.

SentenceBERT

More recent approaches to document embedding attempt to leverage powerful language models like BERT. Reimers and Gurevych 2019 construct an updated version of BERT specifically to tackle document embedding, as BERT has huge overheads when performing document embedding. By introducing siamese and triplet networks, the authors of Sentence-BERT were able to reduce these overheads drastically.

2.2.6 Dimension Reduction

Dimension reduction is the process of reducing a high-dimensional vector to a low-dimensional one, while retaining meaningful properties of the vector. This process is used in a number of cases in AI, including reducing the curse of dimensionality in large sparse vectors, reducing noise or data visualisation.

PCA

Principal Component Analysis (PCA) is a linear data transformation that projects a high dimensional space to a lower dimensional one. It does so while while maximising the preserved variance obtained in the reconstructed lower dimensional space. In practice, the transformation selects the eigenvectors of the

covariance matrix of the data corresponding to the largest eigenvalues[11] (Pearson 1901). PCA is often used as a baseline in dimension reduction contexts.

Autoencoder

Autoencoders are artificial neural networks which use unsupervised learning to create encodings of unlabeled data. For a given space X, an autoencoder can be represented as the composite function of two functions $g \circ f$: an encoder f which maps the original space X to a latent space Z and a decoder g which maps the latent space Z to the original space X (Louppe 2022). These encoding and decoding function can be linear, but more interestingly non-linear. In this case, the encoder and decoder can both be artificial neural networks, such as multilayer perceptrons or convolutional networks.

[11] These eigenvectors with the largest eigenvalues are referred to as the Principal Component, hence the name.

Chapter 3

Related works

In this section we will go over different scientific papers which proposed solutions to coreference resolution task. Coreference resolution can be done in different settings: either within document (WD) or across documents (CD), and either entity coreference resolution or event coreference resolution. Some of these papers propose solution for more than one of these settings, such as the joint approach of H. Lee, Recasens, et al. 2012 which tackles both entity and event coreference resolution in the CD setting. Table 3.1 provides an overview of the different papers discussed here.

A discussion of the prominent datasets used for this task is provided in Section 5.1.

	Within Document	Cross Document
Entity Coreference Resolution	• (K. Lee, He, Lewis, et al. 2017) • (K. Lee, He, and Zettlemoyer 2018) • (Joshi, Levy, et al.2019) • (Joshi, D. Chen, et al. 2020)	
Joint Coreference Resolution		• (H. Lee et al. 2012) • (Barhom et al. 2019) • (Cattan et al. 2021) • (Caciularu et al. 2021)
Event Coreference Resolution		• (Meged et al. 2020) • (Zeng et al. 2020) • (X. Yu, Yin, and Roth 2022)

Table 3.1: Overview of the different papers discussed and the sub-task of coreference resolution they tackle

3.1 WD Coreference Resolution

3.1.1 WD Entity Coreference Resolution

As mentioned previously, most papers tackling coreference resolution do so in a WD setting. The current state-of-the-art in WD event coreference resolution includes Clark and C. D. Manning 2016, K. Lee, He, Lewis, et al. 2017; K. Lee, He, and Zettlemoyer 2018 and Joshi, Levy, et al. 2019; Joshi, D. Chen, et al. 2020.

K. Lee, He, Lewis, et al. 2017 propose their end-to-end coreference model, which is a span-based model[1], meaning it does not perform Mention Detection separately but integrates it as part of coreference resolution. The model therefore considers all spans as possible mentions. For a given span x and possible antecedent spans Y, the model learns a distribution:

$$P(y) = \frac{e^{s(x,y)}}{\sum_{y' \in Y} e^{s(x,y')}} \tag{3.1}$$

The function $s(x, y)$ provides a score between spans x and y. This score is the product between the probability of x being a mention, the probability of y being a mention and the probability of the two spans being coreferring (should they turn out to be mentions). The mention probability is computed as the output of a feedforward neural network. x and y are represented as inputs using fixed-length span representations the contatenation of 2 bi-directional LSTM states of the span endpoints. The LSTM is fed with combination of GLoVe (Pennington, Socher, and C. Manning 2014) and ELMo (Peters et al. 2018).

K. Lee, He, and Zettlemoyer 2018 propose a higher-order coreference model which improves upon K. Lee, He, Lewis, et al. 2017 by upgrading the span representations and adding an attention vector computed over the span's tokens. This inferred vector allows the model to model coreference decisions based on previous coreference decisions.

Joshi, Levy, et al. 2019 take the higher-order coreference model's architecture and replace the LSTM encoder with a BERT transformer. This improves accuracy on the coreference task, especially in the processing of words with similar but distinct meanings (e.g. teacher and professor). This approach is further developed by Joshi, D. Chen, et al. 2020, who design a variant of BERT specially pre-trained to model spans. This model is currently state of the art on the OntoNotes dataset (Weischedel et al. 2013), one of the reference datasets for WD coreference resolution.

[1] A span in NLP is a slice of a document containing an ordered number of tokens.

3.2 CD Coreference Resolution

3.2.1 CD Joint Entity and Event Coreference Resolution

Entity and event coreference resolution were traditionally considered as two separate tasks. This has changed with the works of H. Lee, Recasens, et al. 2012 and Barhom et al. 2019 on joint coreference on the ECB+ dataset. We observe a new interest in the field for solving both entity and event coreference resolution together. More information about the works of H. Lee, Recasens, et al. 2012 and Barhom et al. 2019 can be found in Chapter 4.

Cattan et al. 2021 applies K. Lee, He, Lewis, et al. 2017's end-to-end architecture to the Cross Document setting, using the RoBERTa-large model to encode each document separately and then train the pairwise coreference scorer. Their model is applied to both entity and event coreference resolution.

Caciularu et al. 2021 introduce a new Language Model designed specifically to tackle multi-document tasks. This is done by introducing pretraining over topics instead of single documents, and by employing new forms of transformers able to process much longer sets of inputs. They also replace context embedding of mentions commonly used in most approaches with embeddings of the document in which the mentions are.

3.2.2 CD Event Coreference resolution

Meged et al. 2020 further the work of Barhom et al. 2019 and extend their model using a paraphrase[2] quality scorer to supervise event coreference resolution.

Zeng et al. 2020 and X. Yu, Yin, and Roth 2022 also use the pairwise scorer approach of Barhom et al. 2019, but create the features only based on the output of Language Models and SRLs. Zeng et al. 2020 input two sentences labelled with coreference mentions into the BERT-large Language Model, which is concatenated with the semantic role embedding. These are processed by a Transformer encoder before being processed by the scorer. The resulting model is end-to-end. X. Yu, Yin, and Roth 2022 follows a similar approach, but instead use the RoBERTa-large Language model. The authors also forego the the contextualization tokens of Zeng et al. 2020 when performing pairwise classification.

[2]Paraphrasing is the expression of the same discourse object using different words.

Chapter 4

Methodology

In this chapter we will first review the Original architecture of Barhom et al. 2019, then our suggestions to improve it[1].

4.1 Joint Entity and Event Coreference Resolution

This work extends the work of H. Lee, Recasens, et al. 2012 and Barhom et al. 2019 in the joint coreference resolution of entities and events. This approach to coreference resolution was introduced by Lee et al. in their 2012 paper *Joint Entity and Event Coreference Resolution across Documents*, where they aim to leverage the link between events and entities. This link is based on the fact that events, apart from the action itself, are characterised by 3 elements: time, location and their arguments. These arguments often correspond to discourse entities. As such, knowing that two arguments corefer is helpful for finding coreference relations between events since arguments play a key role in describing an event. In turn, knowing that two events corefer is helpful for finding coreference relations between entities.

In their original paper, H. Lee, Recasens, et al. 2012 iteratively and jointly build clusters of event and entity mentions. Event mentions encodings containing a reference to the entities which fulfill one of the roles of the event. In turn, entity mentions contain a reference to the events they fill a role for. As entity and event mentions are clustered together, the features of the mentions are regenerated and updated.

In practice, the authors first cluster together documents to decrease the search space for cross-document coreference resolution. From these documents they extract entity and event mentions using the Stanford coreference resolution system (Raghunathan et al. 2010; H. Lee, Peirsman, et al. 2011) and enrich entity mentions with information pertaining to lexical, syntactic, semantic, and discourse using deterministic models. The resolution model is based on a linear

[1] We would like to point out that the improvement ideas were the results of suggestions and subsequent discussion with Judicaël Poumay.

Algorithm 1: Outline of H. Lee, Recasens, et al. 2012's Joint Coreference Resolution algorithm

Require: D: document set
R: coreference model
`/* clusters of mentions */`
$E \leftarrow \{\ \}$
`/* clusters of documents */`
$C \leftarrow$ clusterDocuments(D)
for *document cluster* $c \in C$ **do**
  `/* all mentions in one doc cluster: */`
  $M \leftarrow$ extractMentions(c)
  `/* singleton mention clusters: */`
  $E' \leftarrow$ singletonMentions(M)
  while $\exists e_1, e_2 \in E'$ *s.t.* $R(e_1, e_2) > 0.5$ **do**
    $E' \leftarrow$ merge(e_1, e_2, E')
  end
  `/* append to global output */`
  $E \leftarrow E + E'$
  return E
end

regression model, which scores the quality of a merging of clustered entities. Two clusters clusters whose similarity is scored over 0.5 by the linear regression model.

Barhom et al. 2019 furthered this joint coreference resolution approach by combining Lee et al.'s approach with a neural architecture. By using a neural network classifier, word embedding models and a language model, the authors attempt to overcome one of Lee et al.'s limitations: their sparse representations. These representations were built using lexical resources such as WordNet (Miller 1995), which are limited in coverage, and with a context modelled using semantic role dependencies, which do not cover the entire sentential context in most cases. Barhom et al. 2019 instead introduce mention representations using , which are fed pairwise into a Multi Layered Perceptron.

Barhom et al. 2019 instead propose an iterative algorithm which clusters entities and events interdependently in alternation. The final clusters of entities and events are constructed incrementally. The work for this paper builds upon the architecture of Barhom et al. 2019, while adding several key changes.

4.2 Original Model Methodology

Barhom et al. 2019 cluster entities and events together in clusters of coreference. Using an agglomerative clustering algorithm, different mentions (either events or entities) are clustered together based on a similarity metric. This metric is a

pairwise score, provided by a trained model. In this section we will present the different elements of Barhom et al. 2019's solution in detail, as our work builds upon it. For simplicity's sake, we will refer from now on to this model as the Original model.

To provide a pairwise score between two mentions, several steps are needed. We will first discuss the representation of mentions in Section 4.2.1, then how two mentions are combined in Section 4.2.2 and fed to a pairwise scorer described in Section 4.2.3. A overview of the pairwise scoring process can be found in Figure 4.1. The overall algorithms can be found in Algorithms 2 and 3 for training and inference respectively.

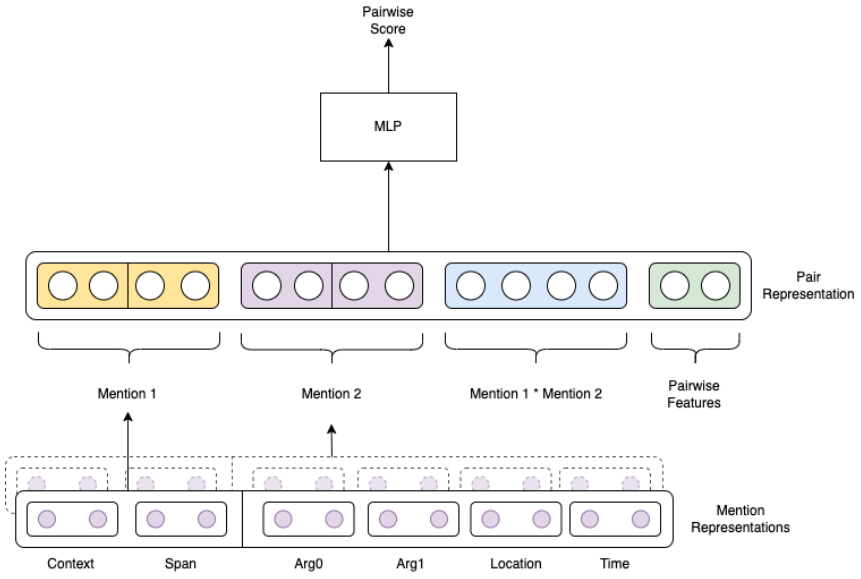

Figure 4.1: An illustration of the pairwise mention scorer from Barhom et al. 2019. The bottom vectors are mention representations. The input to the network is a concatenation of two mention vectors with their element-wise multiplication and additional pairwise features.

4.2.1 Mention Representation

A mention m, be it an event m_v or entity m_e, is represented as the concatenation $\overrightarrow{v}(m)$ of 3 vectors: the span vector $\overrightarrow{s}(m)$, the context vector $\overrightarrow{c}(m)$ and the semantic dependency vector $\overrightarrow{d}(m)$. We will go over each vector separately.

Span Vector

The span vector aims to encode the mention through the words in it, referred to as the span. This is done through the concatenation of the word-level embedding and character-level embedding. Barhom et al. 2019 encode words using the GLoVE model (Pennington, Socher, and C. Manning 2014). The choice of this model among the three most common word embedding models (word2vec, GLoVE and FastText) appears further justified as Poumay and

Ittoo 2021 show GLoVe is the best performer in this architecture. Barhom et al. 2019 chose to use pre-trained word embeddings.

What words are encoded differs between events and entities. For events, the head word is encoded. This refers to the verb that the event centres around. In the event "John burnt himself", this would be the word "burnt". For entities, the average of all the word embeddings of all the words in the mention is taken.

To counteract the issue of out-of-context words [2], the authors extend the word embedding with a character-level encoding obtained through an LSTM (Hochreiter and Schmidhuber 1997). This is complementary to the word embedding, but Poumay and Ittoo 2021 show that the two contain a significant amount of mutual information.

Context Vector

A context vector models the surroundings of the mention. The inclusion of this vector to provide additional information is confirmed by several papers (Clark and C. D. Manning 2016; K. Lee, He, Lewis, et al. 2017; Kenyon-Dean, Cheung, and Precup 2018) who suggest that the compatibility of a mention with other candidate mentions may be indicated by its surroundings.

Barhom et al. 2019 modelled the context of a mention using ELMo (Peters et al. 2018).

Semantic Dependency Vector

Barhom et al. 2019 introduce the semantic dependency vector to model dependencies between event and entity clusters. This is an essential element of their approach, as this provides the link between entities and events during training and inference, thereby generating the joint approach. For a given event mention m_{v_i}, 4 semantic roles of interest: Arg0, Arg1, location, and time (Surdeanu et al. 2007). These are extracted using a semantic role labeller (SRL). Arg0 refers to the closest entity on the left of the event, Arg1 to the closest entity on the right of the event.

When an entity m_{e_j} fills one of m_{v_i}'s semantic roles, then the corresponding semantic role value $\overrightarrow{d}_{sem\_arg}(m_{v_i})$ is set to the average values of the span vectors of the entity's cluster: $\overrightarrow{d}_{sem\_arg}(m_{v_i}) = \frac{1}{|c|}\sum_{m \in c} \overrightarrow{s}(m)$. If no entity fills out this function, the value is set to $\overrightarrow{d}_{sem\_arg}(m_{v_i}) = \overrightarrow{0}$. The idea behind this is to consider that the entity cluster as the proxy for a semantic object. This implies that taking the average of the span vectors should encompass this semantic object.

[2] An out-of-context word is a word not in the word embedding model's vocabulary, which it consequently cannot process. The fastText model as explained in Section 2 is designed to process unknown words through sub-word analysis. GLoVE on the other hand has no inherent way to counteract this.

Symmetrically, the semantic role value for an entity m_{e_j} is set to the average values of the span vectors of the cluster of the event m_{e_j} whose semantic role it fills: $\overrightarrow{d}_{sem\_arg}(m_{v_i})$.

The resulting semantic dependency vector is the concatenation of the four semantic role vectors: $\overrightarrow{d}(m) = \left[\overrightarrow{d}_{Arg0}(m), \overrightarrow{d}_{Arg1}(m), \overrightarrow{d}_{loc}(m), \overrightarrow{d}_{time}(m)\right]$.

4.2.2 Pairwise Mention Representation

To evaluate whether two mentions refer to the same semantic object, Barhom et al. 2019 provide a Pairwise mention representation which will be scored by the pairwise scorer detailed in Section 4.2.3. This pairwise mention representation combines two separate mentions m_i and m_j in a single vector $\overrightarrow{v}_{i,j}$. This vector is the concatenation of the two mention representations, their element-wise product and four pairwise features: $\overrightarrow{p}_{i,j} = \left[\overrightarrow{v}_{m_i}, \overrightarrow{v}_{m_j}, \overrightarrow{v}_{m_i} \circ \overrightarrow{v}_{m_j}, f(i,j)\right]$.

The pairwise features are inspired from H. Lee, Recasens, et al. 2012. These pairwise features are binary indicators of whether each of the four semantic dependency arguments correspond between the two mentions. To increase the signal of each feature, these binary features have been encoded as a 50-dimensional embedding. This is important considering the size of the mention vectors (see Table A.2), the signal could get lost otherwise.

4.2.3 Pairwise Scorer

The Original architecture builds a scoring function $S(m_i, m_j)$ which denotes the probability of coreference between the two input mentions. Given that the span of the outputted score (between 0 and 1), we can view this scorer as a classifier whose score is used. In practice the scoring function is modelled as a feed-forward neural network and takes as input the pairwise mention representation detailed in Section 4.2.2. Two separate functions S_V and S_E are trained for events and entities respectively.

4.3 Thresholding

As detailed in the above section, Barhom et al. 2019 perform joint modelling by switching between entity and event training. Nonetheless they alternate only twice between the two models (for a total of 2 sets of 2 training sessions for each model). This intuitively felt like an insufficient leverage of the joint feature of the model, arguably one of its main features. As such our first modification to Barhom et al. 2019 is to increase the number of alternations between entity and event models.

A second intuition concerned the clustering threshold. As mentioned above, training consists of phases of training the pairwise scorer and then of clustering of coreference mentions (events or entity). Barhom et al. 2019 use a threshold of 0.5 when merging clusters in all cases. Our suggestion instead is to set a higher

Algorithm 2: (Barhom et al. 2019) Train

Require: D: document set
M^e, M^v: gold entity/event mentions
T: gold topics (document clusters)
$\{E_t\}_{t \in T}$: gold within-doc entity clusters
$G(\cdot)$: gold mention to cluster assignment

for $t \in T$ **do**
  $V_t \leftarrow$ SingletonEvents(t, M_v)

  while $\exists$ *meaningful cluster-pair merge* **do**
    `/* Entities */`
    $E_t \leftarrow$ UpdateJointFeatures(V_t)
    $S_E \leftarrow$ TrainMentionPairScorer(E_t, G)
    $E_t \leftarrow$ MergeClusters(S_E, E_t)
    `/* Events */`
    $V_t \leftarrow$ UpdateJointFeatures(E_t)
    $S_V \leftarrow$ TrainMentionPairScorer(V_t, G)
    $V_t \leftarrow$ MergeClusters(S_V, V_t)
  end
  return S_E, S_V
end

threshold initially, and then progressively lower it. This intuitively ensures that only the best merges of clusters are done initially, and those of lower quality are only done with the better trained pairwise scorer.

Combining these two solutions we propose to increase the number of alternations, while having a stricter merge threshold and progressively lowering it.

4.4 Document Embedding

4.4.1 Enriching Mention Representations

Mention representations in (Barhom et al. 2019) are decomposed into 3 sections: their span vectors, composed of the word embedding of the head work and the character-level embedding, a context vector provided by ELMo and a semantic vector, which relates to the cluster in which the mention is placed. While the context of a mention does give a summary of the surrounding words and mentions in the sentence, the mention vectors do not contain any direct reference to the document in which the mention is. This intuitively feels like a drawback, as our focus is cross-document coreference resolution and as such, some knowledge of which document the mention is in seems a valuable indication to the pairwise scorer. As such we extend the mention vector by adding a new vector with the embedding of the document the mention is in.

Algorithm 3: (Barhom et al. 2019) Inference

Require: D: document set
M^e, M^v: gold entity/event mentions
$S_E(\cdot,\cdot)$: pairwise entity mention scorer
$S_V(\cdot,\cdot)$: pairwise event mention scorer

$T \leftarrow$ ClusterDocuments(D)
for $t \in T$ **do**
  $V_t \leftarrow$ SingletonEvents(t, M_v)
  $E_t \leftarrow$ PredWithinDocEntityCoref(t, M_e)
  while $\exists$ *meaningful cluster-pair merge* **do**
    `/* Entities */`
    $E_t \leftarrow$ UpdateJointFeatures(V_t)
    $E_t \leftarrow$ MergeClusters(S_E, E_t)
    `/* Events */`
    $V_t \leftarrow$ UpdateJointFeatures(E_t)
    $V_t \leftarrow$ MergeClusters(S_V, V_t)
  end
  return $\{E_t\}_{t\in T}$,$\{V_t\}_{t\in T}$
end

To properly research this train of thought, we test out 3 different document embedding methods. The first is a classic baseline in document embedding averaging the word embeddings of the document. This has been shown to be a very strong baseline (Lau and T. Baldwin 2016). We use the same word embeddings as used elsewhere in the model, GLoVe for convenience and knowing that it performs better than the other traditional word embedding techniques (fastText and word2vec) in this task as shown by (Poumay and Ittoo 2021). The second model is the original document embedding, (Le and Mikolov 2014)'s Paragraph Vectors. Finally we test a state-of-the-art pretrained model, SentenceBERT (Reimers and Gurevych 2019). While initially intended only for sentences, the model is able to take in documents of limited length. After verification, the largest document in the datasets contains 247 tokens, short of the limits of available pretrained models (which range between 256 and 512 tokens as upper bounds).

4.4.2 Enriching Semantic Arguments

While directly enriching the mention vectors with document embeddings seemed like a good idea, we wanted to push this idea to other parts of the model. Specifically, we thought that the semantic argument vectors, based solely on the average of the span, were insufficiently rich. We therefore try to enrich these vectors with the ELMO context vectors. As an additional step, we combine this

approach with that of subsection 4.4.1: we add document embedding to the mention representation vector, but also to the argument vectors. We decide to use the document embedding model producing the best results from subsection 4.4.1.

4.5 Scorer Input Reduction

As mentioned in Section 4.2.2, the input of the MLP assigning the pairwise score is the concatenation of 2 mention representation vectors, their element-wise multiplication and additional pairwise features. This vector is quite large, the authors of the original paper worked with mention representation of size 2774: the concatenation of the context vector of size 1024, 5 GLoVE word embeddings of size 300 and 5 character-level embeddings of size 50. Multiplying this number by 3 (the two mentions and their element-wise multiplication) and adding the 3 additional features of size 50 each, the input size is of 8522. The MLP has 2 hidden layers: the first of size 8522/2 and the second of size 8522. The output is a single number, being the pairwise score. These numbers can be found in A.2.

This input is quite large, leading to a strikingly wide and shallow neural network. (Poumay and Ittoo 2021) performed a comprehensive study of the information of this large input. Their conclusion is that this vector contains a high amount of mutual information. An additional observation is that approximatively two-thirds of the training time is spent training the MLP. All of these problems are further exacerbated by our extensions of the mention vectors in section 4.4. Mention representation vectors and pairwise representation vectors are significantly larger, as can be observed in Tables A.3 and A.4, which increases the training and inference time of the pairwise scorer, and as such also the clustering time. We therefore decided to investigate the compression of the input of the MLP, in other words the pairwise mention representation vector. The aim is to leverage the mutual information in the input vector to compress it as losselessly as possible, with the added improvement of being able to work with a smaller network which can be trained faster.

2 different ways of compressing the input were implemented:

- Reduce the size of the mention representation using PCA. We decided to work on reducing the size of the mention representation, which should in consequence decrease the size of the pairwise representations.
- Introduce an autoencoder to compress the input. As with PCA, we work on reducing the mention representation vector.

In the process of building the autoencoder, we observed that a pairwise representation was mostly a repetition of the span vector of the mention embeddings. Indeed, the only part of a mention embedding which is not composed of span vectors is the context vector, which is not included in the semantic dependency vectors. We want to encode the smallest possible uniform brick of the

embeddings. We therefore make the choice of encoding the span and context vectors. This leads to an issue with the semantic dependency vector, which only contains span vectors without the context. For the auto-encoding, the semantic dependency vectors are modified: instead of only taking the span vector average of the coreference cluster, the average of both the span and context vectors. This will allow us to train a single autoencoder on the span and context autoencoder vectors.

Chapter 5

Implementation Details and Experimental Setup

5.1 Data

In this work the ECB+ dataset is used. Introduced by Cybulska and Vossen 2014, it contains both Within Document Coreference (WDCR) and Cross-document Coreference annotations and is the reference dataset for Cross-Document Coreference tasks. Events and entities are both annotated separately in the dataset, which allows one to tackle a task while ignoring the other. ECB+ is an extension of the EECB dataset, introduced by H. Lee, Recasens, et al. 2012 to perform joint event and entity coreference resolution. EECB, and by extension ECB+, are both extensions of the Event Coreference Bank (ECB) dataset (Bejan and Harabagiu 2014), which contains English-language Google News documents clustered into topics and annotated for event coreference. This dataset was extended by H. Lee, Recasens, et al. 2012 to entities, in order to assess their joint model. ECB+ builds upon EECB by adding several topics concerning a different event for each topic, thereby making the task more difficult but richer.

Conversely, Stylianou and Vlahavas 2021; J. Lu and Ng 2018; Sukthanker et al. 2020 provide an in-depth review of the different coreference resolution datasets. Poumay and Ittoo 2021 explain why these other datasets do not suit the approach taken by Barhom et al. 2019 and consequently ours.

5.2 Data Preprocessing

We naturally follow the setup of Barhom et al. 2019, who themselves follow that of (Cybulska and Vossen 2015). This setup was used by most other papers tackling the same task on ECB+, such as (Kenyon-Dean, Cheung, and Precup 2018). This setup uses a subset of the annotations which has been validated for correctness by Cybulska and Vossen 2014 and allocates a larger portion of the

dataset for training (see Table A.1). Since the ECB+ corpus only annotates a part of the mentions, the setup only uses the gold-standard event and entity mentions and does not require specific treatment.

(Barhom et al. 2019) divide the dataset into three sections: training, development and validation. A breakdown of the composition of each dataset provided by the authors can be found in Table A.1.

5.3 Implementation

Unless otherwise detailed, the training protocol follows that of Barhom et al. 2019. Our implementation differs from the original authors' in that we used updated libraries. We will therefore report it as a separate case in our results. We also do not train for the same number of epochs due to hardware limitations. For reference, Barhom et al. 2019 trains for 50 epochs, while we manage to do 10 in most cases.

5.3.1 Thresholding

We train in alternation entity and event pairwise scorers. After each training phase, clusters are merged with thresholds which are progressively lowered. We start at an initial threshold of 0.9, and we lower by 0.1 after each merge phase, until 0.5 is reached. Unlike Barhom et al. 2019, we only do a single pass for each threshold level. This totals to 5 training phases for each of the models. Once our pairwise scorer is trained, agglomerative clustering is performed and clusters are merged with all candidate singletons and clusters which score above the threshold.

5.3.2 Document Embedding

For the baseline GLoVE average and SentenceBERT, no training is required as both of these are pretrained models. For GLoVE we use the same model as for Barhom et al. 2019 used for word embedding, with an embedding size of 300. For SentenceBERT we use a pretrained model provided by Reimers and Gurevych 2019[1]. In particular we use the *all-mpnet-base-v2* model. This model is labelled as the best performing on encoding among those provided, and is described as being an all-rounder. Its maximum sequence length is of 384, being larger than the biggest document in ECB+.

Our Doc2Vec model on the other hand has been trained on the training dataset. This was shown to provide great results even with limited data (Lau and T. Baldwin 2016). We go through a process of model selection to tune the hyperparameters of the architecture. The evaluation is done on 3 annex tasks. While they are not directly linked to our model's way of working, we created them with the intuition that, having to reach similar objectives, the resulting

[1] `https://www.sbert.net/`

document embedding model contains meaningful information for the task at hand.

1. Classification is performed on the documents, classifying them with their gold topics. We use the SGD classifier [2] from the scikit-learn library with the default hyperparameters. This model was selected as it was reasonably simple while providing a probability output and not a binary output, which we want for our model selection criteria. We evaluate classification using top-3 accuracy. The intuition behind this is that several topics are quite similar and we did not want to penalise excessively models unable to differentiate very similar events. Additionally, the gold-label for topics are not hand-crafted but generated using clustering as described in Barhom et al. 2019. The results from Barhom et al. 2019 show that these document clusters are of high quality.

2. The same task is done with an ExtraTrees classifier [3] from the scikit-learn library, with the same evaluation procedure and the default hyperparameters.

3. Clustering is performed with a KMeans classifier. The number k of clusters is set to the number of topics. Evaluation is done using the V-measure[4] (Rosenberg and Hirschberg 2007), which allows us to compare the output of a clustering and the gold labels of a classification.

The output of all 3 classifiers is averaged and gives a unique evaluation score for a given document embedding model.

To select the correct parameters, a grid search is performed. For each task we perform K-fold model validation with $K = 20$ and repeated it five times. As advised by literature, we did not choose the number of epochs through the grid-search and arbitrarily set it at 10. The model is trained on the Training part of the dataset and evaluated on the Development section of the dataset. The resulting hyperparameters can be found in Table A.5. Of particular note is the fact that we obtained a vector size of 150.

5.3.3 Pairwise Scorer

The scorer is an feed-forward artificial neural network (or MLP) with 2 hidden layers: one half the size of the input layer and the other the same size as the input layer. The input layer is set to the size of the pairwise mention representation vector and the output to a single number. The ReLU Nair and Geoffrey E Hinton 2010 activation is used for the hidden layers, and the output is run through a sigmoid. Layer sizes are further detailed in Section 4.5. S_E (resp. S_V) is trained based on all the pairs of entity (resp. event) mentions

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeClassifier.html`

[4] The V-measure computes the mutual information between two sequences of labellings.

that belong to different entity (resp. event) clusters in the current predicted configuration E_t. The gold label is set to 1 if the two entities belong to the same coreference chain, and 0 otherwise. Binary cross entropy is used as a loss function for training.

5.3.4 Input Reduction

One of our first observations were that using an autoencoder reduced the time spent in training the scorer, but this was balanced out by the time spent to train the autoencoder. To improve time performance, we attempted to train the autoencoder separately from the rest of the model. To do this we would need to know in advance what the model inputs are. This is straightforward in almost all cases: our gold mention embeddings are known in advance and embedding models (word, context and document) are pre-trained. The only exception is the character-level LSTM, which is trained at training time. We therefore train the character-level LSTM at the same time as the autoencoder. The training parameters are otherwise similar to that of Barhom et al. 2019, with a hidden layer of size 50.

As a baseline for encoding the inputs we use PCA and in particular its scikit-learn implementation of the algorithm [5]. To ensure that the comparison is fair as a baseline, PCA is applied to mention embeddings with the static features. The output size for the dimension reduction is set to 256.

In building the autoencoder, we thought we could enrich the training signal beyond the reconstruction error. Considering the objective of the task in which the autoencoder will feature, we chose to use the encoding layer to train a classifier. This classifier classifies the mention representation extract to its corresponding gold coreference cluster. Since we choose to use a differentiable classifier, the training signal from the classifier can be summed with the autoencoder's normal training signal and sent back into the encoder section of the autoencoder to enrich the training. Our intuition is that this signal will increase similarity between the encoded mentions belonging to the same gold cluster, thus making them easier to classify as similar down the line ; or at the very least help to preserve in the encoded mentions some of the information to distinguish mentions from different coreference clusters.

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

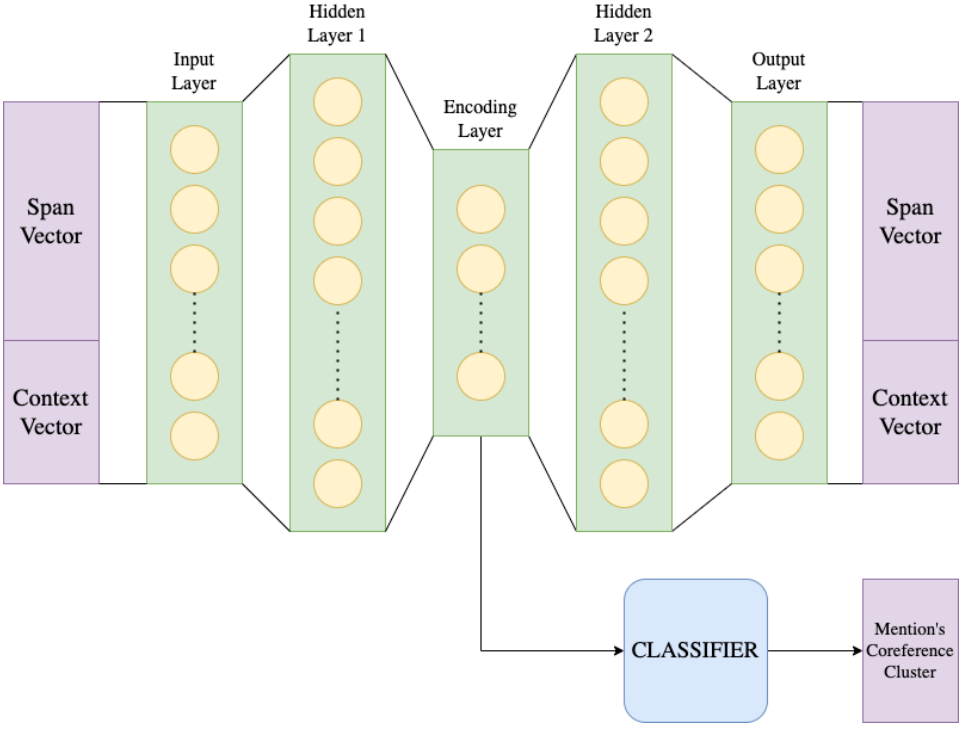

Figure 5.1: Illustration of our autoencoder training architecture

In practice we built a simple classifier using the PyTorch library, an artificial neural network with 2 hidden layers of size 512. Rectified Linear Unit is used as activation layer, and cross entropy loss as a loss function. The architecture of the autoencoder is illustrated in Figure 5.1.

For the autoencoder, we built a model using the PyTorch library. The Adam optimizer is used. To select the correct parameters for our autoencoder model, we use a similar approach as with our Doc2vec training. We chose an arbitrary encoding size of 256. A series of different models with different hyperparameters were trained on the training dataset and evaluated on the three tasks described in Section 4.4.1. The model scoring the highest is selected, and the results of the grid search can be found in Table A.6.

5.3.5 Evaluation Metrics

We evaluate our coreference resolution models using the CoNLL scorer (Pradhan et al. 2014). With the scorer, we evaluate the resulting models on the four most popular evaluation of the six metrics provided:

- MUC (Vilain et al. 1995), which compares the number of links in the gold coreference cluster (G) to the number of links missing in the predicted coreference clusters. This is done using the number of partitions of the gold cluster through the predicted ones (P). Given a gold cluster G, its number of internal links ($|G|-1$) and the number of missing links ($|p(G)|$), the recall of MUC can be computed as: $R = \frac{(|K|-1)-(|p(K)|-1)}{|K|-1} = \frac{|K|-|p(K)|}{K|-1}$

- B^3 (Bagga and B. Baldwin 1998), for which precision for a mention m_i is the number of correct mentions in the gold coreference clusters that contain the mention, divided by the total number of mentions in the gold cluster. Recall on the other hand is the number of correct mentions in the gold cluster divided by the number of mentions in the gold clusters which contains mention m_i. The overall precision and recall for the entire document is the weighted sums of precision and recall of the individual mentions.
- CEAF_e (Luo 2005) is one of the two variants of the CEAF metric. CEAF computes the best alignment of subsets of gold clusters and predicted clusters. For any mapping $a \in A_m$ the total similarity $\Phi(a)$ is the sum of all similarities. The best alignment a^* is found by maximizing the sum of similarities $\Phi(a)$ between the gold and predicted clusters, while the maximum total similarity is the sum of the best similarities. Precision and recall are computed as such: $P = \frac{\Phi(a^*)}{\sum_i \phi(P_i,P_i)}$ $R = \frac{\Phi(a^*)}{\sum_i \phi(G_i,G_i)}$. For CEAF_e, the similarity measure is $\phi_e(G,P) = \frac{2|G \cap P|}{|G|+|P|}$
- CoNLL F_1 (Pradhan et al. 2014), the average of the first three metrics.

There has been debate about the use of these metrics in the past, notably concerning their interpretability (Luo 2005) and the apparent lack of agreement between the three metrics. Holen 2013 perfoms an in-depth study of the metrics, and shows the low correlation between all the metrics in the task (in particular MUC and CEAF_e have a correlation going down to as low as 0.22).

5.4 Experimental setup

Like Barhom et al. 2019, we follow Cybulska and Vossen 2015's experimental setup. This setup uses a subset of the annotations which has been validated for correctness by Cybulska and Vossen 2014 and allocates a larger portion of the dataset for training (see Table A.1). Since the ECB+ corpus only annotates a part of the mentions, the setup uses the gold-standard event and entity mentions and does not require specific treatment for unannotated mentions during evaluation. We also use the SwiRL model (Surdeanu et al. 2007), and follow Barhom et al. 2019's heureustics for increasing coverage.

Chapter 6

Results

In this chapter, we will go over the baselines and evaluated models. We will then present the results of our tests in Tables 6.2 and 6.2, and analyse them. Finally we will discuss possible areas of further investigation.

6.1 Baselines

To evaluate our results, we compare our results to the latest state-of-the-art models as baselines. We use the following models for both entity and event coreference resolution:

- Barhom et al. 2019
- Cattan et al. 2021
- Caciularu et al. 2021

We use the following models as baselines for event coreference resolution only:

- Meged et al. 2020
- Zeng et al. 2020

6.2 Evaluated Models

We run the following models:

0. The **Original** model proposed by Barhom et al. 2019“
1. **Thresholds**: The Original model with progressive threshold lowering and increased switches between event and entity embedding.
2. **Document Embedding**: The Original model with mention encodings extended with the embedding of their document. Three variations are used:

(a) **GloVe Averaging**

(b) **Doc2Vec**

(c) **SentenceBERT**

3. **Extended Semantic Dependecy Vector**:

   (a) **ELMo**: The Original model where the Semantic dependency vectors are extended with the context vector (ELMo)

   (b) **SentenceBERT**: Model 3.C with only the Semantic dependecy vectors are extended with the document embedding vector (SentenceBERT)

4. **Input Reduction**:

   (a) **PCA + ELMo**: Model 3.A with mention encoding reduction using PCA

   (b) **PCA + ELMo + SentenceBERT**: Model 3.C with mention encoding reduction using PCA

   (c) **Autoencoder + ELMo**: Model 3.A with mention encoding reduction using an autoencoder

		MUC			B^3			$CEAF_e$			CoNLL
		R	P	F_1	R	P	F_1	R	P	F_1	F_1
	Barhom et al. 2019	78.6	80.9	79.7	65.5	76.4	70.5	65.4	61.3	63.3	71.2
	Cattan et al. 2021 - Gold	85.7	81.7	83.6	70.7	74.8	72.7	59.3	67.4	63.1	73.1
	Caciularu et al. 2021 - CDLM	88.1	91.8	89.9	82.5	81.7	82.1	81.2	72.9	76.8	**82.9**
0	Original (Barhom et al. 2019)	66.3	84.4	74.5	55.8	85.8	67.6	75.0	50.4	60.2	67.4
1	Thresholds	75.7	81.3	78.4	62.5	78.0	69.4	65.2	55.9	60.3	69.3
2.A	DocEmb: GLoVe Averaging	77.3	83.9	80.5	64.6	78.4	70.8	70.4	59.3	64.4	71.9
2.B	DocEmb: Doc2Vec	79.3	81.5	80.4	64.4	75.6	69.5	64.6	60.7	62.6	70.9
2.C	DocEmb: SentenceBERT	77.8	82.9	80.3	65.4	77.3	70.9	68.5	59.9	63.9	71.7
3.A	SemDep: ELMo	78.7	81.2	79.9	64.6	76.1	69.8	63.8	59.7	61.6	70.4
3.B	SemDep: SentenceBERT	77.9	81.3	79.5	64.9	76.8	70.4	64.5	58.7	61.4	70.4
4.A	InRed: PCA + ELMo	71.4	78.6	74.8	59.5	78.1	67.5	63.4	52.1	57.2	66.5
4.B	InRed: PCA + ELMo + SentenceBERT	37.5	77.8	50.6	40.7	89.8	56.0	77.6	34.7	48.0	51.5
4.C	InRed: Autoencoder + ELMo	56.2	83.0	67.0	47.7	87.4	61.8	76.9	43.5	55.5	61.4

Table 6.1: Results on entity cross-document coreference resolution on ECB+ test set.

		MUC			B^3			$CEAF_e$			CoNLL
		R	P	F_1	R	P	F_1	R	P	F_1	F_1
	Barhom et al. 2019	77.6	84.5	80.9	76.1	85.1	80.3	81.0	73.8	77.3	79.5
	Meged et al. 2020	78.8	84.7	81.6	75.9	85.9	80.6	81.1	74.8	77.8	80.0
	Cattan et al. 2021 - Gold	85.1	81.9	83.5	82.1	82.7	82.4	75.2	78.9	77.0	81.0
	Zeng et al. 2020	85.6	89.3	87.5	77.6	89.7	83.2	84.5	80.1	82.3	84.6
	Caciularu et al. 2021 - CDLM	87.1	89.2	88.1	84.9	87.9	86.4	83.3	81.2	82.2	**85.6**
0	Original (Barhom et al. 2019)	73.6	83.7	78.3	73.6	85.8	79.2	80.8	70.6	75.3	77.6
1	Thresholds	72.3	84.1	77.8	72.0	86.8	78.7	81.4	69.5	75.0	77.1
2.A	DocEmb: GLoVe Averaging	72.7	83.7	77.8	72.9	86.9	79.3	81.6	70.4	75.6	77.6
2.B	DocEmb: Doc2Vec	78.9	82.1	80.4	76.9	83.2	79.9	77.8	74.3	76.0	78.8
2.C	DocEmb: SentenceBERT	84.5	80.1	82.2	81.3	81.0	81.1	73.3	78.6	75.8	79.7
3.A	SemDep: ELMo	78.0	83.0	80.4	74.3	84.9	79.2	79.1	73.9	76.4	78.7
3.B	SemDep: SentenceBERT	78.1	83.6	80.7	76.5	84.8	80.4	79.5	73.6	76.4	79.2
4.A	InRed: PCA + ELMo	82.5	73.7	77.8	79.3	72.2	75.6	63.8	74.5	68.7	74.0
4.B	InRed: PCA + ELMo + SentenceBERT	57.3	75.1	65.0	64.0	83.0	72.3	76.4	59.3	66.8	68.0
4.C	InRed: Autoencoder + ELMo	54.8	87.1	67.3	62.6	91.9	74.5	86.8	59.9	70.9	70.9

Table 6.2: Results on event cross-document coreference resolution on ECB+ test set.

6.3 Analysis

We note that when running the Original model of Barhom et al. 2019 (0), whether with specified and updated libraries and minor modifications, we never reach the quality of results described by Barhom et al. 2019 or other baseline models. It is important to note that we did not train the model for 50 epochs like Barhom et al. 2019 as the cost would be prohibitive prohibitive. Whatsmore we observe a fast convergence during training (4 epochs were usually sufficient for most models to reach their best scores). We therefore trained the models in our own setting for around 10 epochs.

Another observation during training was that there seemed to be a significant variance in training performances depending on the seed chosen for the clustering operations. As such a proper evaluation of these algorithms should be undertaken through a stability analysis (Ben-David, von Luxburg, and Pál 2006).

The Threshold model (1) shows a slight increase in performance in entities coreference resolution and a slight decrease with events. From the mediocre changes, we can gather the following two lines of thoughts. First, that slowly lowering the clustering threshold to incite it towards better clusterings does not seem to have a noticeable effect. Second, that Barhom et al. 2019's models make near full use of the joint information that is present between events and entities and that the amount of improvement that can be gained from increasing it is not worth the time investment (training the threshold model (1) took almost three times what it took with the Original model (0)).

Looking at the Original model (0), the Thresholds model (1) and the GLoVe Averaging model (2.A), the trend between the results for entity and event coreference resolution clearly differ. While for entities we observe a net improvement in performances between the extended and the Original models, there is little to no change for events. We can hypothesise that this is due to underspecified mention encodings for entities, while those of events are saturated: adding a little information to the encodings leads to improvement for entities and little to none in events. This intuition is reinforced by the results of the baselines: they show an improvement of 6.1 points in F_1 CoNLL score between Barhom et al. 2019 and Caciularu et al. 2021 when tackling events and almost twice this amount (11.7 points) in entities.

When comparing the different document embedding extensions (2.A, 2.B, 2.C), we observe that the heavier models perform marginally better than the lighter ones, but that trend is not followed with entities. Instead, Doc2Vec (2.B) underperforms compared to the GLoVe averaging document embedding technique (2.A). GloVe averaging (2.A) turns out to be the best performing model among those we computed for entities. The addition of more complex document embeddings does seem to impact positively the events.

Extending the semantic dependency vectors does seem to provide some minor improvements, be it with ELMo (3.A) or SentenceBERT (3.B). This does come at a substantial cost in time, almost twice as long to train, as the vectors are a bit larger (see Table A.4). Additionally the increased used of language models also impacts training time negatively.

As for input reduction, we notice as expected a drop in performance. PCA + ELMo (4.A), for example, loses 0.9 points for entity coreference resolution and 3.5 points for event coreference resolution against the Original model (0). This is quiet impressive, considering that the mention embeddings get reduced to less than 10% of their original size using PCA. In particular we note its high recall We note that PCA, as a baseline, has no knowledge of the task that we will be trying to perform and simply optimizes for reconstruction error. These observations can also be put in correlation with the work of Poumay and Ittoo 2021 and their analysis of the significant amount of mutual information in the mention encodings. On the other hand, PCA + ELMo + SentenceBERT (4.B) performs very poorly and and loses 16 and 10 points for entities and events respectively. We explain this by the significant increase in mention size without augmenting the encoding size or the amount of data. This loss of performance can also be observed in the Mean Squared Error in Table A.7. We expected the autoencoder + ELMo model (4.C) to perform better than the simpler baselines as its more complex architecture and training should give it an edge on PCA. This turned out not to be the case, as the autoencoder + ELMo model (4.C) performed more poorly than PCA + ELMo (4.A). We hypothesize this being due to the insufficient amount of training data to properly train an autoencoder. To obtain better results, additional mention data would need to be leveraged from other datasets.

6.4 Further developments

While studying Barhom et al. 2019's modelling, we found several aspects of the mention representation as potentially improvable. We list them below:

- In our analysis of the results, we aimed not to over-interpret these results, due to the low interpretability of the metrics on one hand, and due to the lack of stability on the other. Further interpretation of the results would require an error analysis as performed by Barhom et al. 2019. The idea is to sample some of the errors in the coreference clusters and observe what is the source of this error.
- To simplify the problem, the experimental setup is such that coreference resolution is performed separately within each topic. This does simplify a complex problem, but is realistic so long as the topic clustering is very accurate. It also does not consider the fact that documents can contain multiple topics. A path of investigation could therefore be to address the problem of coreference resolution on the same ECB+ dataset in a single all-encompassing topic.
- As mentioned in Section 6, there seems to be a significant amount of instability in this problem. As such a stability analysis of the different models would provide a better picture of their actual performances.
- An entity mention span is represented as the average of all the word embeddings of the mention. This problem can be set as a document embedding problem. While we have discussed that averaging word embeddings is a strong baseline for document embedding, there are stronger methods as we have shown in this paper. Replacing the average word embedding by another document embedding technique could lead to a more meaningful embedding.
- In the course of this work, we noticed that the pairwise scorer's MLP architecture is badly optimized. More care could be taken in building and training this model to ensure that it performs precisely and in a time efficient manner.
- The use of the SwiRL SRL library and ELMo have been pointed out by other papers as outdated (Caciularu et al. 2021). As such replacing these libraries with more modern tools, notably Transformer based Language Models.
- While the autoencoder has shown poor results in this task, we suspect that this is due to a lack of data. Training the autoencoder with more data from other datasets, in particular the larger Within Document coreference resolution ones, could prove fruitful.

Chapter 7

Conclusion

In this paper we developed the approach of H. Lee, Recasens, et al. 2012 and Barhom et al. 2019 towards the joint resolution of entity and event coreference in a cross document setting. We showed that Barhom et al. 2019's joint model already makes near full use of the joint information between entity and event coreference resolution by increasing the amount of switches between the two models.

Our addition of document embeddings to the mention encoding has shown improvement on the entity coreference embedding, regardless of the complexity of the document embedding technique. Finally we have shown that harsh compression of the mention representation using simple baselines can still provide good results. Furthermore, we observed that the ECB+ dataset contains insufficient data to effectively train an autoencoder.

Bibliography

ACE (Automatic Content Extraction) English Annotation Guidelines for Events (2005). 5.4.3 2005.07.01. Manual.

Bagga, Amit and Breck Baldwin (1998). "Entity-Based Cross-Document Coreferencing Using the Vector Space Model". In: *Proceedings of the 36th Annual Meeting on Association for Computational Linguistics* -. Vol. 1. Montreal, Quebec, Canada: Association for Computational Linguistics, p. 79. DOI: 10.3115/980845.980859.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (May 2016). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv: 1409.0473 [cs, stat].

Barhom, Shany et al. (June 2019). "Revisiting Joint Modeling of Cross-document Entity and Event Coreference Resolution". In: *arXiv:1906.01753 [cs]*. arXiv: 1906.01753 [cs].

Baum, Leonard E. and Ted Petrie (Dec. 1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". In: *The Annals of Mathematical Statistics* 37.6, pp. 1554–1563. ISSN: 0003-4851. DOI: 10.1214/aoms/1177699147.

Bejan, Cosmin and Sanda Harabagiu (June 2014). "Unsupervised Event Coreference Resolution". In: *Computational Linguistics* 40.2, pp. 311–347. ISSN: 0891-2017, 1530-9312. DOI: 10.1162/COLI_a_00174.

Bellman, Richard (1984). *Dynamic Programming*. Princeton, NJ: Princeton Univ. Pr. ISBN: 978-0-691-07951-6.

Ben-David, Shai, Ulrike von Luxburg, and Dávid Pál (2006). "A Sober Look at Clustering Stability". In: *Learning Theory*. Ed. by David Hutchison et al. Vol. 4005. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 5–19. ISBN: 978-3-540-35294-5 978-3-540-35296-9. DOI: 10.1007/11776420_4.

Bengio, Y. (2009). "Learning Deep Architectures for AI". In: *Foundations and Trends® in Machine Learning* 2.1, pp. 1–127. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000006.

Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *JOURNAL OF MACHINE LEARNING RESEARCH* 3, pp. 1137–1155.

Bojanowski, Piotr et al. (June 2017). "Enriching Word Vectors with Subword Information". In: *arXiv:1607.04606 [cs]*. arXiv: 1607.04606 [cs].

Caciularu, Avi et al. (Sept. 2021). *CDLM: Cross-Document Language Modeling*. arXiv: 2101.00406 [cs].

Cattan, Arie et al. (June 2021). *Cross-Document Coreference Resolution over Predicted Mentions*. arXiv: 2106.01210 [cs].

Cho, Kyunghyun et al. (Sept. 2014). "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv:1406.1078 [cs, stat]*. arXiv: 1406.1078 [cs, stat].

Chomsky, Noam (2015). *Syntactic Structures*. Repr. der Ausg. 's-Gravenhage, Mouton,1957. Mansfield Centre, CT: Martino Publ. ISBN: 978-1-61427-804-7.

Clark, Kevin and Christopher D. Manning (2016). "Improving Coreference Resolution by Learning Entity-Level Distributed Representations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 643–653. DOI: 10.18653/v1/P16-1061.

Cybulska, Agata and Piek Vossen (May 2014). "Using a Sledgehammer to Crack a Nut? Lexical Diversity and Event Coreference Resolution". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

— (June 2015). "Translating Granularity of Event Slots into Features for Event Coreference Resolution." In: *Proceedings of the the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Denver, Colorado: Association for Computational Linguistics, pp. 1–10. DOI: 10.3115/v1/W15-0801.

Dai, Andrew M., Christopher Olah, and Quoc V. Le (July 2015). *Document Embedding with Paragraph Vectors*. arXiv: 1507.07998 [cs].

Deloche, François (June 2017a). *A Diagram for a One-Unit Long Short-Term Memory (LSTM)*.

— (June 2017b). *Structure of RNN*.

Deng, Li and Yang Liu, eds. (2018). *Deep Learning in Natural Language Processing*. New York, NY: Springer Berlin Heidelberg. ISBN: 978-981-10-5208-8.

Devlin, Jacob et al. (May 2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs].

Eisenstein, Jacob (2019). *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-04284-0.

Elman, Jeffrey L. (Mar. 1990). "Finding Structure in Time". In: *Cognitive Science* 14.2, pp. 179–211. ISSN: 03640213. DOI: 10.1207/s15516709cog1402_1.

Firth, J. R. (Nov. 1935). "THE TECHNIQUE OF SEMANTICS." In: *Transactions of the Philological Society* 34.1, pp. 36–73. ISSN: 0079-1636, 1467-968X. DOI: 10.1111/j.1467-968X.1935.tb01254.x.

Fleuret, Francois (2021). *Recurrent Neural Networks*. École Polytechnique Fédérale de Lausanne.

Goldberg, Yoav (2017). *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies #37. San

Rafael, Calif.: Morgan & Claypool Publishers. ISBN: 978-1-62705-298-6 978-1-68173-235-0. DOI: 10.2200/S00762ED1V01Y201703HLT037.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-03561-3.

Group, The Stanford Natural Language Processing (2022). *Coreference Resolution*.

Harris, Zellig S. (Aug. 1954). "Distributional Structure". In: *Word* 10.2-3, pp. 146–162. ISSN: 0043-7956, 2373-5112. DOI: 10.1080/00437956.1954.11659520.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

Holen, Gordana Ilić (June 2013). "Critical Reflections on Evaluation Practices in Coreference Resolution". In: *Proceedings of the 2013 NAACL HLT Student Research Workshop*. Atlanta, Georgia: Association for Computational Linguistics, pp. 1–7.

Ittoo, Ashwin (Nov. 2019). *Word Embeddings*. HEC Management School – University of Liège.

Joshi, Mandar, Danqi Chen, et al. (Jan. 2020). *SpanBERT: Improving Pre-training by Representing and Predicting Spans*. arXiv: 1907.10529 [cs].

Joshi, Mandar, Omer Levy, et al. (2019). "BERT for Coreference Resolution: Baselines and Analysis". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 5802–5807. DOI: 10.18653/v1/D19-1588.

Jurafsky, Dan and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed. Prentice Hall Series in Artificial Intelligence. Upper Saddle River, N.J: Pearson Prentice Hall. ISBN: 978-0-13-187321-6.

Kamath, Uday, John Liu, and James Whitaker (2019). *Deep Learning for NLP and Speech Recognition*. Cham: Springer. ISBN: 978-3-030-14596-5 978-3-030-14595-8 978-3-030-14598-9.

Karttunen, Lauri (1969). "Discourse Referents". In: *Proceedings of the 1969 Conference on Computational Linguistics* -. Sång-Säby, Sweden: Association for Computational Linguistics, pp. 2–2. DOI: 10.3115/990403.990487.

Kenter, Tom, Alexey Borisov, and Maarten de Rijke (2016). "Siamese CBOW: Optimizing Word Embeddings for Sentence Representations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 941–951. DOI: 10.18653/v1/P16-1089.

Kenter, Tom and Maarten de Rijke (Oct. 2015). "Short Text Similarity with Word Embeddings". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. Melbourne Australia:

ACM, pp. 1411–1420. ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806475.

Kenyon-Dean, Kian, Jackie Chi Kit Cheung, and Doina Precup (2018). "Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization". In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1–10. DOI: 10.18653/v1/S18-2001.

Kiros, Ryan et al. (June 2015). *Skip-Thought Vectors*. arXiv: 1506.06726 [cs].

Kostadinov, Simeon (Feb. 2019). *Encoder-Decoder Sequence to Sequence Model*.

Lau, Jey Han and Timothy Baldwin (July 2016). *An Empirical Evaluation of Doc2vec with Practical Insights into Document Embedding Generation*. arXiv: 1607.05368 [cs].

Le, Quoc V. and Tomas Mikolov (May 2014). "Distributed Representations of Sentences and Documents". In: *arXiv:1405.4053 [cs]*. arXiv: 1405.4053 [cs].

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (May 2015). "Deep Learning". In: *Nature* 521.7553, pp. 436–444. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539.

Lee, Heeyoung, Yves Peirsman, et al. (June 2011). "Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task". In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 28–34.

Lee, Heeyoung, Marta Recasens, et al. (July 2012). "Joint Entity and Event Coreference Resolution across Documents". In: vol. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, pp. 489–500.

Lee, Kenton, Luheng He, Mike Lewis, et al. (2017). "End-to-End Neural Coreference Resolution". In: DOI: 10.48550/ARXIV.1707.07045.

Lee, Kenton, Luheng He, and Luke Zettlemoyer (Apr. 2018). *Higher-Order Coreference Resolution with Coarse-to-fine Inference*. arXiv: 1804.05392 [cs].

Lopez Medel, Maria (July 2021). "Gender Bias in Machine Translation: An Analysis of Google Translate in English and Spanish". In: *Academia Letters*. ISSN: 2771-9359. DOI: 10.20935/AL2288.

Louppe, Gilles (Aug. 2022). *Lecture 10: Auto-encoders and Variational Auto-Encoders*. Online.

Lu, Jing and Vincent Ng (July 2018). "Event Coreference Resolution: A Survey of Two Decades of Research". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, pp. 5479–5486. ISBN: 978-0-9992411-2-7. DOI: 10.24963/ijcai.2018/773.

— (2020). "Conundrums in Entity Coreference Resolution: Making Sense of the State of the Art". In: *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, pp. 6620–6631. DOI: 10.18653/v1/2020.emnlp-main.536.

Lu, Yaojie et al. (Feb. 2022). "End-to-End Neural Event Coreference Resolution". In: *Artificial Intelligence* 303, p. 103632. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103632.

Luo, Xiaoqiang (2005). "On Coreference Resolution Performance Metrics". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 25–32. DOI: 10.3115/1220575.1220579.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). *Effective Approaches to Attention-based Neural Machine Translation*. arXiv: 1508.04025 [cs].

Meged, Yehudit et al. (Nov. 2020). "Paraphrasing vs Coreferring: Two Sides of the Same Coin". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 4897–4907. DOI: 10.18653/v1/2020.findings-emnlp.440.

Mikolov, Tomas, Kai Chen, et al. (Sept. 2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]*. arXiv: 1301.3781 [cs].

Mikolov, Tomas, Ilya Sutskever, et al. (Oct. 2013). *Distributed Representations of Words and Phrases and Their Compositionality*. arXiv: 1310.4546 [cs, stat].

Miller, George A. (Nov. 1995). "WordNet: A Lexical Database for English". In: *Communications of the ACM* 38.11, pp. 39–41. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/219717.219748.

Nair, Vinod and Geoffrey E Hinton (Jan. 2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In.

Natural Language (2021).

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan (2002). "Thumbs up?: Sentiment Classification Using Machine Learning Techniques". In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02*. Vol. 10. Not Known: Association for Computational Linguistics, pp. 79–86. DOI: 10.3115/1118693.1118704.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (Feb. 2013). "On the Difficulty of Training Recurrent Neural Networks". In: *arXiv:1211.5063 [cs]*. arXiv: 1211.5063 [cs].

Pearson, Karl (Nov. 1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572. ISSN: 1941-5982, 1941-5990. DOI: 10.1080/14786440109462720.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

Peters, Matthew E. et al. (Mar. 2018). *Deep Contextualized Word Representations*. arXiv: 1802.05365 [cs].

Poumay, Judicael and Ashwin Ittoo (Oct. 2021). *A Comprehensive Comparison of Word Embeddings in Event & Entity Coreference Resolution*. arXiv: 2110.05115 [cs].

Pradhan, Sameer et al. (2014). "Scoring Coreference Partitions of Predicted Mentions: A Reference Implementation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 30–35. DOI: 10.3115/v1/P14-2006.

Pustejovsky, James et al. (Mar. 2004). "TimeML: Robust Specification of Event and Temporal Expressions in Text." In: *New Directions in Question Answering*. Ed. by Mark T. Maybury. AAAI Press, pp. 28–34. ISBN: 1-57735-184-3.

Raghunathan, Karthik et al. (Oct. 2010). "A Multi-Pass Sieve for Coreference Resolution". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, pp. 492–501.

Reimers, Nils and Iryna Gurevych (Aug. 2019). *Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks*. arXiv: 1908.10084 [cs].

Rosenberg, Andrew and Julia Hirschberg (June 2007). "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 410–420.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088, pp. 533–536. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/323533a0.

Stylianou, Nikolaos and Ioannis Vlahavas (Apr. 2021). "A Neural Entity Coreference Resolution Review". In: *Expert Systems with Applications* 168, p. 114466. ISSN: 09574174. DOI: 10.1016/j.eswa.2020.114466. arXiv: 1910.09329 [cs].

Sukthanker, Rhea et al. (July 2020). "Anaphora and Coreference Resolution: A Review". In: *Information Fusion* 59, pp. 139–162. ISSN: 15662535. DOI: 10.1016/j.inffus.2020.01.010.

Surdeanu, M. et al. (June 2007). "Combination Strategies for Semantic Role Labeling". In: *Journal of Artificial Intelligence Research* 29, pp. 105–151. ISSN: 1076-9757. DOI: 10.1613/jair.2088. arXiv: 1110.0029 [cs].

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (Dec. 2014). "Sequence to Sequence Learning with Neural Networks". In: *arXiv:1409.3215 [cs]*. arXiv: 1409.3215 [cs].

Trunk, G. V. (July 1979). "A Problem of Dimensionality: A Simple Example". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.3, pp. 306–307. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766926.

Vaswani, Ashish et al. (Dec. 2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs].

Vilain, Marc et al. (1995). "A Model-Theoretic Coreference Scoring Scheme". In: vol. Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995. MUC.

Weischedel, Ralph et al. (Oct. 2013). *OntoNotes Release 5.0*. DOI: 10.35111/XMHB-2B84.

Yu, Juntao, Bernd Bohnet, and Massimo Poesio (June 2020). *Neural Mention Detection*. arXiv: 1907.12524 [cs].

Yu, Lei et al. (Dec. 2014). *Deep Learning for Answer Sentence Selection*. arXiv: 1412.1632 [cs].

Yu, Xiaodong, Wenpeng Yin, and Dan Roth (Jan. 2022). *Pairwise Representation Learning for Event Coreference*. arXiv: 2010.12808 [cs].

Zabokrtsky, Zdenek (2016). *Feature Engineering in Machine Learning*. Online.

Zeng, Yutao et al. (2020). "Event Coreference Resolution with Their Paraphrases and Argument-aware Embeddings". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 3084–3094. DOI: 10.18653/v1/2020.coling-main.275.

Appendix A

Tables

	Train	Validation	Test	Total
# Topics	25	8	10	43
# Sub-topics	50	16	20	86
# Documents	574	196	206	976
# Sentences	1037	346	457	1840
# Event mentions	3808	1245	1780	6833
# Entity mentions	4758	1476	2055	8289
# Event chains	1527	409	805	2741
# Entity chains	1286	330	608	2224

Table A.1: ECB+ statistics (including singleton clusters). The topics are split in the following manner: - train: 1, 3, 4, 6-11, 13-17, 19-20, 22, 24-33; validation: 2, 5, 12, 18, 21, 23, 34, 35; test: 36-45. (Barhom et al. 2019)

Original Model Vector Sizes				Size
Mention Embedding	Span Vector	Word Embedding	GLoVe	300
		Character Level Embedding	LSTM	50
	Context Vector	Context Embedding	ELMo	1024
	Semantic Features		Span Vector	350
			Span Vector	350
			Span Vector	350
			Span Vector	350
Sum				**2774**
Pairwise Features		Mention Embedding x3		8322
		Arg0		50
		Arg1		50
		loc		50
		time		50
	Sum			200
MLP Input				8522

Table A.2: Breakdown of the Vector Sizes for the Original model from Barhom et al. 2019

ID	Model	Length	Mention Size	Pairwise Vector
2.A	GloVe Average	300	3074	9422
2.B	Doc2Vec	150	2924	8972
2.C	SentenceBERT	768	3542	10826

Table A.3: Vector Sizes for models using Document Embeddings

ID	Model	Length	Semantic Feature	Mention Size	Pairwise Vector
3.A	ELMo	1024	1374	6870	11594
3.B	SentenceBERT	768	1118	6614	10826

Table A.4: Vector Sizes for models extending the Semantic Dependency Vectors

Parameter	Value
doc2vec variant	DBOW
dm	0
hs	1
min_count	4
vector size	150
window	6

Table A.5: Result of the hyperparameter search for the doc2vec document embedding.

Parameter	Value
batch size	64
encoding layer dimension	256
epochs	100
hidden layer dimension	8192
learning rate	0.0001
number of hidden layers	1

Table A.6: Result of the hyperparameter search for the autoencoder.

Dimension Reduction Technique	Extended Semantic Dependency Vector	Coreference Setting	MSE
PCA	ELMo	entity	0.0236
PCA	ELMo	event	0.0203
Autoencoder	ELMo	entity	0.0263
Autoencoder	ELMo	event	0.0227
PCA	ELMo + SentenceBERT	entity	0.0297
PCA	ELMo + SentenceBERT	event	0.0247
Autoencoder*	ELMo + SentenceBERT	entity	0.0331
Autoencoder*	ELMo + SentenceBERT	event	0.0283

Table A.7: Mean Squared Error for dimension reduction models. Note that the final two autoencoders were never tested in the context of the task.