

Heidelberg University
Faculty of Modern Languages
Department of Computational Linguistics

BACHELOR THESIS
B.A. in Computational Linguistics

Neural Resolution
of Comparative Anaphora

AUTHOR

Victor Zimmermann

SUPERVISOR

Prof Dr Katja Markert

MATRICULATION NO.

3460308

SECOND ASSESSOR

Prof Dr Michael Herweg

Heidelberg, 5th December 2019

Abstract

This work presents the first application of a neural network architecture to the task of comparative anaphora resolution. To this end, a corpus of 512 in-text samples of nominal and named entity antecedents from multiple domains was created, as well as an automatically collected corpus of 3825 coreference mention pairs for pre-training.

Relevant terms and concepts of anaphora and recurrent neural networks, as well as related work in the field of anaphora resolution are introduced.

Two hypotheses are tested using a BiLSTM and the new comparative anaphora corpus: (1.) The semantic properties of word embeddings should encode the information relevant to the non-trivial (i.e. non-syntactic) samples in the test set. (2.) Since other anaphora resolution fields provide larger data sets than the comparative anaphora task, the similarity between tasks can be employed to pre-train a model before fine-tuning on the main task.

The experiments performed with this setup largely suffer from the size of the test data, even with cross-validation, as less data leads to higher thresholds for significance. However, all naïve baselines were surpassed by multiple tested models by a significant margin, building a solid neural baseline for future systems.

The provided error analysis does give insights into how a mixed-domain corpus affects the success rate, as well as the impact of implicit classification constraints.

Abriss

Diese Arbeit präsentiert die erste Anwendung einer neuronalen Netzwerkarchitektur für die Aufgabenstellung der komparativen Anaphorikresolution. Zu diesem Zwecke wurde ein Korpus, bestehend aus 512 textinternen Beispielen von nominalen und Eigennamen-Antezedenten, sowie ein maschinell aggregiertes Korpus aus 3825 koreferenten Erwähnungspaaren zum Vortraining, erstellt.

Relevante Begriffe und Konzepte der Anaphorik, rekurrenter neuronaler Netze, sowie der verwandten Arbeiten im Gebiet der Anaphorikresolution werden eingeführt.

Zwei Hypothesen werden mittels eines bidirektionalen Langkurzzeitgedächtnisnetzwerkes und des neuen komparativen Anaphorikkorpus geprüft: (1.) Die semantischen Eigenschaften von Worteinbettungen sollten für nicht-triviale (d.h. nicht-syntaktische) Beispiele im Testdatensatz relevante Informationen encodieren. (2.) Aufgrund der Tatsache, dass andere Gebiete der Anaphorikresolution größere Datensätze zur Verfügung stellen, kann die Ähnlichkeit der Aufgabenstellungen genutzt werden, um ein Modell vorzubilden, ehe es für die Hauptaufgabenstellung feineingestellt wird.

Die mit dieser Konfiguration durchgeführten Experimente leiden größtenteils unter der Menge der Testdaten, selbst mit Kreuzvalidierung, da weniger Daten zu einer höheren Signifikanzschwelle führen. Indessen konnten sämtliche naiven Vergleichssysteme von mehreren getesteten Modellen geschlagen werden, was sie als solide neuronale Vergleichssysteme für zukünftige Experimente auszeichnet.

Die vorliegende Fehleranalyse gewährt Einblicke in den Einfluss eines Korpus verschiedener Domänen auf die Erfolgsrate, wie auch die Auswirkung von impliziten Klassifikationseinschränkungen.

Acknowledgements

To Trang, who, when all was lost,
let me sleep on her bedroom floor.

Table of Contents

Abstracts	ii
English	ii
German	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
List of Acronyms	ix
1 Introduction	1
2 Theoretical Background	4
2.1 Anaphora	4
2.2 Neural Machine Learning	5
2.2.1 Feedforward Neural Network	5
2.2.2 Recurrent Neural Networks	7
2.2.3 (Bidirectional) Long Short-Term Memory	8
2.2.4 Vector Semantics & Embeddings	12
3 Related Work	14
3.1 Anaphora Resolution	14
3.1.1 Comparative Anaphora Resolution	14
3.1.2 Coreference Resolution	15
3.1.3 Bridging	15
3.1.4 Pre-Training	16
4 Corpus	17
4.1 Preprocessing & Filtering	17
4.1.1 Pre-Training	18
4.2 Annotation	19
4.3 Analysis	19
5 Model	21
5.1 Input & Output	21
5.2 Embedding Layer	23
5.3 BiLSTM Layer	23
5.4 Feed Forward Layer	23

Table of Contents

5.5	Loss & Back-Propagation	24
6	Experiments & Results	25
6.1	Evaluation Measure	25
6.2	Parameter Tuning	26
6.3	Baselines	27
6.3.1	Random	27
6.3.2	Recency	27
6.3.3	Head Match.	27
6.4	Results	27
6.4.1	Ablation	28
6.4.2	Training Data	29
6.5	Error Analysis	31
6.5.1	Overlap	31
6.5.2	Non-Head Predictions	31
6.5.3	Part-of-Speech.	32
6.5.4	Source Domain	33
6.5.5	Comparative Modifier	34
6.5.6	Distance	35
6.6	Significance	35
7	Conclusion	37
8	Future Work	39
	Bibliography	40
A	Notes on Annotation.	45
B	Notes on Implementation	46

List of Tables

1	Referenced Activation Functions	6
2	Data Set Statistics	20
3	Input/Output Example	22
4	Cross-Validation Fold Sizes.	25
5	Experiments for Parameter Tuning	26
6	Macro- and Micro-Averaged Success Rates	28
7	Ablation Experiments	28
8	Macro- and Micro-Averaged Success Rates After Ablation	29

List of Figures

1	Elman RNN	7
2	Bidirectional RNN	9
3	RNN Cell	10
4	LSTM Cell	11
5	Domain Proportions	20
6	Model Diagram	21
7	Success Rate at Sample Size	30
8	Prediction Overlap	31
9	Number of Non-Head Classifications at Sample Size	32
10	Success Rate for Parts-of-Speech	32
11	Success Rate for Domains	33
12	Success Rate for Comparative Modifiers	34
13	Success Rate at Distance	35
14	Bootstrapped Confidence Intervals	36
15	Interface for Annotation.	46

List of Acronyms

BiLSTM	bidirectional long short-term memory network
BoW	bag of words
CBOW	continuous bag of words
CD	cardinal number
ECTB	English-Chinese Parallel Treebank
FFNN	feedforward neural network
GloVe	Global Vectors for Word Representation
GRU	gated recurrent unit
JJ	adjective
JJR	adjective, comparative
LSA	latent semantic analysis
LSTM	long short-term memory network
NE	named entity
NLP	natural language processing
NLTK	natural language toolkit
NN	noun, singular
NNP	proper noun, singular
NNPS	proper noun, plural
NNS	noun, plural
NP	noun phrase
NT	New Testament

List of Acronyms

POS	part-of-speech
PRP	pronoun, personal
ReLU	rectified linear unit
RNN	recurrent neural network
softmax	normalised exponential function
tanh	hyperbolic tangent
tf-idf	term frequency–inverse document frequency
WP	wh-pronoun, personal
WSJ	Wall Street Journal

1. Introduction

This work deals with the resolution of comparative anaphora using a neural architecture. To this purpose I collected a data set of 512 samples of in-text comparative anaphora relations, with the heads of both anaphor and all possible (candidate) antecedents annotated. The system uses word embeddings and a long short-term memory network (LSTM) to embed words within their context and returns the index of the most likely antecedent out of a list of possible antecedents given the anaphor index. I also assess the impact of pre-training on data extracted automatically from the related coreference resolution task, for which previous work has made more sizeable corpora available. Each system is evaluated against three naïve baselines.

Comparative anaphora resolution, as used in this thesis, refers to the identification of antecedents to referential noun phrases with non-pronominal heads and comparative modifiers.¹

The ship is in the harbour now, see if you can spot *him*.

Another immigrant, comin' up from the bottom.

In the example above, *another immigrant* is referring to *him*, but (in-text) possible antecedents also include *the ship*, *the harbour* or *you*. Deciding which of these phrases is the true antecedent can be considered the resolution task. This thesis deals with non-trivial instances of comparative anaphora. The mention pruning therefore largely mirrors Modjeska (2003): Syntactic antecedents (i.e. other-than and list constructions) have been discarded for this task, as their identification can be solved on a syntactic, rather than a semantic basis. Idiomatic expressions have also been discarded, as have been reciprocal expressions (e.g. “each other”) and phrases referring to previous discourse (e.g. “in other words”). As a consequence the data set presented here largely concerns itself with nominal and pronominal antecedents, with some few samples of cardinal numbers and adjectives. Comparative anaphors can of course also refer to verbal phrases (e.g. “He *biked* to Hanoi. *Other vehicles* were available.”), although this is infrequent and was not annotated for this task. The task only considers antecedents within the same or

¹Any concrete realisation of comparative anaphora hereafter is referred to as *anaphor* (pl. *anaphors*), while the term *anaphora* is used when referring to the concept of anaphora at large.

1. Introduction

preceding sentence, which is due to logistic limitations and in line with previous setups (Modjeska, 2003). The process of corpus creation is discussed in more detail in Section 4.

Formally the task can be defined as follows: For an anaphor a with its head at index i_a , find index $i_{\bar{c}}$ of any true antecedent $\bar{c} \in \bar{C}_a$, with set of true antecedent entity mentions $\bar{C}_a \subseteq C_a$, from set of candidate antecedents C_a .

Comparative anaphora resolution is closely tied to other forms of anaphora resolution, notably metonymy (Markert and Hahn, 2002). It stands to reason, that more understanding of a task where bridging, metaphor and metonymy have a high impact, like comparative anaphora resolution, can also lead to more understanding in these fields.

Question and answering systems that accept unrelated queries are one application for comparative anaphora resolution, as users may ask for *other* information on a previous subject, like in normal conversation:

Q: How many articles are there in the Federalist Papers?

A: 85 articles.

Q: How many did Jay and Madison write?

A: John Jay got sick after writing five. James Madison wrote twenty-nine.

Q: Who wrote the other fifty-one?

This last question requires the system to understand that fifty-one refers to the subset of the Federalist Papers that were not written by John Jay and James Madison, which was mentioned by name only two questions ago.

Comparative anaphors can also encode new information, e.g. in the phrase “Washington and other presidents”. Here the anaphor introduces the information that Washington is a president, which can be utilised for information extraction. Of course, if the antecedent is not linked to the anaphor like in this example, resolving the anaphor often becomes dependent on knowing the relation beforehand. “He talked to Washington. Other presidents where more reclusive.” can only be resolved with prior knowledge.

In this thesis the following hypotheses are explored: (1.) Since the comparative anaphora resolution task, as evaluated on the corpus presented here, focuses on the resolution of relationships that require semantic or world knowledge, a feature conveying semantic information, like the word embeddings used as input for the neural systems in this thesis, should perform well. (2.) Because deep learning techniques require large data sets to train, pre-training on tasks with more data

1. Introduction

available could be a method to circumvent the data sparsity problem encountered in niche subjects.

I that multiple neural models, given enough data, outperform all three baselines significantly. However, because of the limited data size, a significant difference between the systems could not be established. It should be highlighted that the system trained only on data extracted from coreference samples performed reasonably well, considering it has not seen any actual comparative anaphora data during training.

Section 3 highlights the related work in comparative anaphora resolution, as well as other anaphora resolution tasks like coreference resolution or bridging, following which Section 2 provides a brief review of the theoretical background of the systems used for the resolution task. The data used for training is described in Section 4 with details on the annotation process and the extraction of pre-training samples. The system itself is discussed in-depth in Section 5 and evaluated in Section 6. A summary and conclusion of the presented work is given in Section 7 and finally further venues for research on this topic are discussed in Section 8.

2. Theoretical Background

2.1. Anaphora

Anaphora describes the dependence of two discourse elements, of which one, the **anaphor** can only be interpreted fully in context of the **antecedent**. Consider the following example:

The measure will end 20 years of review, and guarantees *China* the low tariffs *almost all other nations* receive.

Almost all other nations needs the antecedent *China* to be coherent, as it is defined by the exclusion of one particular nation.

One well studied subsection of anaphora resolution is coreference (Carter, 1987; Deemter and Kibble, 2000). Two mentions are said to be coreferent if both refer to the same global entity.

The package was termed excessive by the Bush administration , but *it* also provoked a struggle with influential California lawmakers [...].

It is coreferent with *the package*, but *the Bush administration* is syntactically also possible. Coreference and comparative anaphora are related in the fact that both mentions are connected by some sort of similarity, for coreference, mentions are similar in the fact that they share the same *entity*. In comparative anaphora both mentions are members of the same *group*. This is most evident for other-anaphora, since the shared group is most often spelled out in the anaphor. *China* and *other nations* both are subsets of some *nation* group.

Another major subset of anaphora resolution research was done in the field of bridging resolution. Bridging as a whole refers to discourse-new anaphora dependent on previous context. Bridging, albeit differently defined by different researchers, can be broken down into two sub-tasks (Baumann, 2012; Roesiger et al., 2018). *Referential* bridging refers to cases where the anaphor is dependent on an antecedent to be coherent.

The linguistics lecture is delivered by an external researcher. *The reading list* will be made available shortly.

Without the context of a university lecture, it is not clear what exactly the reading list is referring to. Comparative anaphora almost always falls into this definition

2. Theoretical Background

and can be considered part of referential bridging.

On the other hand, *lexical* bridging, as defined by Baumann (2012) refers to meronymy, hyponymy and other lexical semantic relations, where the mentions stand in some kind of *conceptual* relation.

The parliament voted against the motion, much to the delight of *the opposition*.

It is often the case, as in the example above, that both referential and lexical bridging appear for the same mention pair, although both bridging notions express different concepts. It is evident from the part-whole relationship between *opposition* and *parliament* that the opposition within said parliament is referenced. There is at once the lexical relation that the opposition, as part of parliament voted on the motion, as well as the referential relation that the opposition belongs to *this* parliament.

2.2. Neural Machine Learning

2.2.1. Feedforward Neural Network

Feedforward neural networks (FFNNs) are statistical models that ostensibly mimic the makeup of the human brain by chaining multiple layers of perceptrons (Rosenblatt, 1958), which allows the neural network to solve non-linear problems like the XOR-Problem (Goodfellow et al., 2016). With vector representations (see Section 2.2.4) these models have successfully been employed in nearly every field of natural language processing (NLP), including coreference (Section 3.1.2) and bridging resolution (Section 3.1.3).

FFNNs build on the regression model of a standard perceptron and expand it through *hidden layers* of neurons, which are just perceptrons with associated input, weights and output vectors. Instead of calculating the output from the input directly, each neuron's output is further fed into the next layer as input, thus in a standard, fully-connected FFNN a neuron's *hidden state* is calculated from the output of the previous layer's hidden states multiplied by the neuron's weight vector. Each layer is usually equipped with an activation function, which normalises the output of each neuron. The activation functions relevant to the works discussed here are listed in Table 1.

The full process outlined above is what is referred to as a *forward pass*, a side effect of which is that the whole process can be expressed as a series of matrix operations. This allows us to express the states of a hidden layer h as the matrix product of its

2. Theoretical Background

weights ω and the input vector x (as well as some bias b and activation function σ):

$$h = \sigma(\omega \cdot x + b)$$

A cell's input is not dependent on its output, hence why this architecture is called feedforward.

Name	Function	Derivative
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$\frac{\partial f(x)}{\partial x} = f(x)(1 - f(x))$
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\frac{\partial f(x)}{\partial x} = 1 - f(x)^2$
Softmax	$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{ x } e^{x_j}}$ for $i = 1, \dots, x $	$\frac{\partial f(x)_i}{\partial x_j} = f(x)_i(\delta_{ij} - f(x)_j)$

Table 1: Activation functions referenced in this thesis, as well as their derivatives, with Kronecker delta δ_{ij} ².

The weight matrix for each layer is updated through a process called backward differentiation, which in the case of neural networks is referred to as back-propagation (Rumelhart et al., 1986). Back-propagation is largely based on the chain rule from differential calculus, applied to a loss function. More specifically one needs to calculate the partial derivative of the loss function with respect to each parameter. As the computation of each hidden layer is entirely based on the previous layer, this can be expressed as such a chain calculation. For example, for a two-layer neural network with weight matrices $\omega_{1,2,3}$, the calculation of the output \tilde{y} from input x , with activation functions σ, ς may look like this:

$$\begin{aligned}
 h_1 &= \sigma(\omega_1 \cdot x + b_1) \\
 h_2 &= \omega_2 \cdot h_1 + b_2 \\
 \tilde{y} &= \varsigma(h_2) \\
 \Leftrightarrow \tilde{y} &= \varsigma(\omega_2 \cdot \sigma(\omega_1 \cdot x + b_1) + b_2)
 \end{aligned}$$

In most modern frameworks for deep learning architectures, each function is implemented with its own derivative function. In a process called auto-differentiation a stack of all called functions during the forward pass is kept and the updates applied accordingly. Before each parameter can be updated, we need to define a loss function ($L(\tilde{y}, y)$) over the output, and a learning rate (λ). The goal in choosing

² $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{else.} \end{cases}$

2. Theoretical Background

a loss function is to approximate how close the network output is to the desired result. The learning rate dampens the impact of each update, ideally prohibiting over-correction. The general update function for a parameter ω can therefore be defined as:

$$\omega_{new} = \omega_{old} - \frac{\partial L(\tilde{y}, y)}{\partial \omega_{old}} \cdot \lambda$$

This is done iteratively over multiple training samples. Each cycle over the training data is called an epoch. Bundling samples and applying average weight updates is called mini batch gradient descent, the number of samples bundled is also a meta parameter, batch size. This is a compromise between stochastic gradient descent, where an update is applied after each sample, and batch gradient descent, where the gradient is calculated over the entire data set and an update is applied once per epoch.

2.2.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) borrow many concepts from FFNNs, with the notable exception that a cell is impacted by previous outputs beyond back-propagation. This *recurrence* allows neural networks using a variant of this architecture to model time-dependent variables. In its most basic version (Elman, 1990), RNNs feed the hidden layer h_{t-1} of the previous time step as input into the current hidden layer h_t , as is shown in Figure 1.

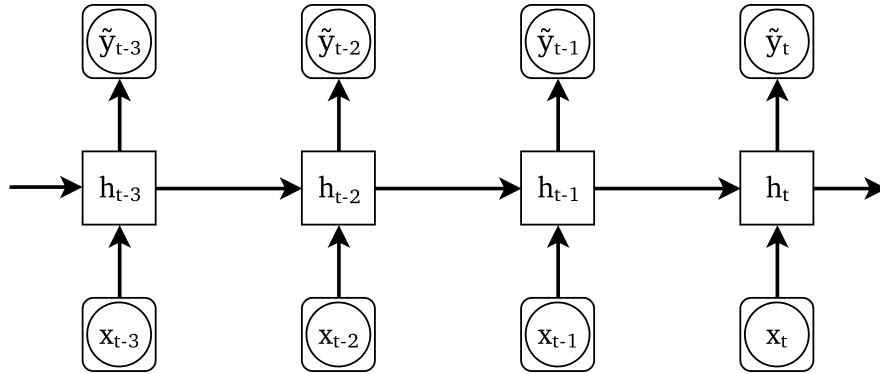


Figure 1: Simple Elman RNN. In addition to each time step input x_t , the previous hidden state h_{t-1} is used to calculate the system output \tilde{y} at t .

Forward inference is handled in a similar manner to a FFNN, with previous hidden state h_{t-1} , weight vectors v , ν and ω , and activation functions σ and ς :

$$h_t = \sigma(\nu \cdot h_{t-1} + \omega \cdot x_t)$$

$$\tilde{y}_t = \varsigma(v \cdot h_t)$$

2. Theoretical Background

Note that the output function also features an additional weight vector, which helps to deal with the 2-ary function of the hidden state to both inform the next cell and provide an output at time step.

When it comes to back-propagation, this 2-ary function presents a problem, as not only the output at the current time step produces an error, which has to be applied, but also every following cell. The error term δ , i.e. how much of the loss can be attributed to a particular parameter, for each hidden state can be stated as the sum of each function's error rate, with cell state z , before applying an activation function:

$$\begin{aligned}\delta_{out} &= L' \sigma'(z) \\ \delta_h &= \sigma'(z) \cdot v \cdot \delta_{out} + \delta_{next}\end{aligned}$$

Updating the output's weight function v only requires the error rate δ_{out} , as it is not fed any previous cell state.

Each weight matrix can thus be updates as follows:

$$\begin{aligned}\frac{\partial L}{\partial v} &= \delta_{out} \cdot h_t \\ \frac{\partial L}{\partial v} &= \delta_h \cdot x_t \\ \frac{\partial L}{\partial \omega} &= \delta_h \cdot h_{t-1}\end{aligned}$$

Just like with FFNNs, updates can be applied using the call stack and moderated using a learning rate.

2.2.3. (Bidirectional) Long Short-Term Memory

For the general neural network architecture, activation functions are usually selected for a specific task. For LSTMs the sigmoid and hyperbolic tangent (tanh) functions are employed for their definitional properties, so for the following sections, σ and ς no longer denote arbitrary activation functions, but the sigmoid and tanh function respectively.

A major weakness of the standard RNN architecture are long-distance dependencies, as each cell state is modified by the following cells and gets more mangled the more cells it passes. One approach to not lose the information from the start of a sentence, is to run the network both ways, making it a *bidirectional* RNN (see Figure 2). For classification, the cell states of the left-to-right and the right-to-left RNN are combined. The combined cell states are especially helpful when the whole

2. Theoretical Background

input, e.g. a tokenised sentence or a multi-token span, needs to be classified. In this case, the last cell state of each directed RNN is concatenated and passed to the classifier.

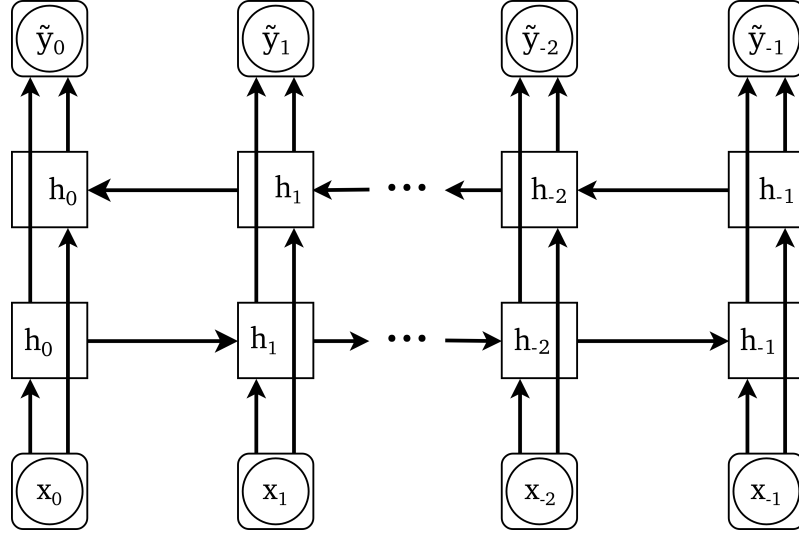


Figure 2: A bidirectional RNN. Input at time step is passed into two RNNs doing their forward passes in opposite directions. The outputs are concatenated and used for classification.

With this extended architecture it becomes even more useful that the output is given its own weight vector and activation function, as the input for calculating \tilde{y}_t no longer solely depends on one cell state h_t , but two.

Still, with bidirectionality, RNNs perform poorly on long-dependency tasks. The two main extensions of the core RNN architecture, LSTMs (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs) by Cho et al. (2014) try to deal with this issue by expanding the cell through a more complex internal structure. We will only look at the LSTM cell, as the GRU architecture is basically a simplified LSTM.

2. Theoretical Background

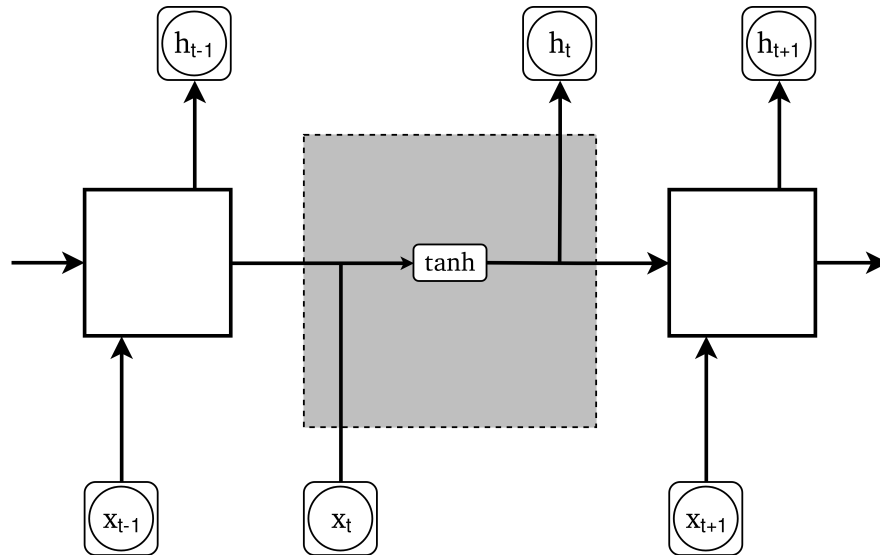


Figure 3: A standard RNN cell. Each cell behaves like layer in a FFNN, but sharing weights and activation functions.

For RNNs, the cell provides little more than an activation function, \tanh in the case of Figure 3. The added complexity becomes apparent when looking at the bowels of the LSTM cell in Figure 4. Here so called *gates* allow the network to remember, but also to forget certain information. Each gate consists of a sigmoid neural layer and a pointwise multiplication operation. The 2-ary functionality of the hidden state and new input in RNNs is now split between cell and hidden state. Both are still given to the the following cell as input, but they serve a different purpose. The cell state is constructed to convey long-term information, while the hidden state represents the cell's output, as before.

2. Theoretical Background

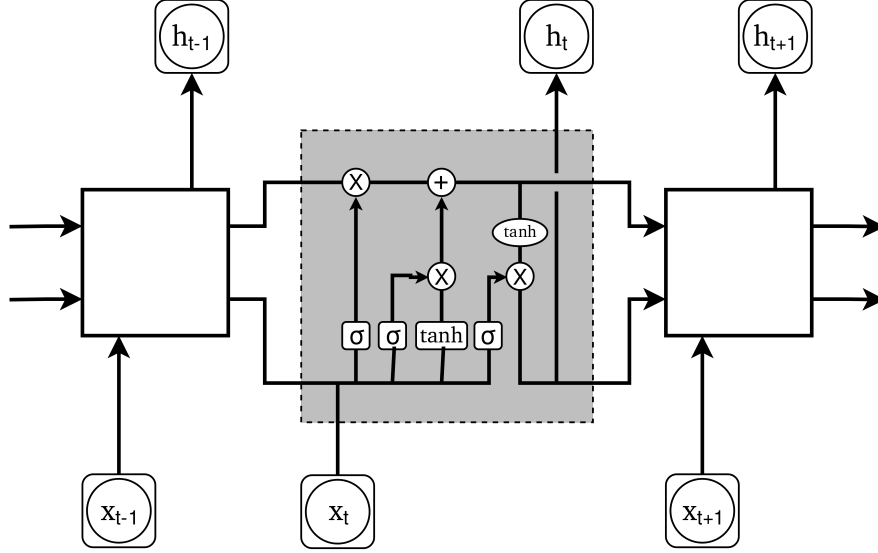


Figure 4: An LSTM cell after Hochreiter and Schmidhuber (1997). Sigmoid functions and multiplication operators make up so-called *gates*. Within the cell, circles denote element-wise operators, squircles indicate single-layer neural networks, converging arrows concatenations and diverging arrows multiple recipients of the same vector (i.e. copying).

The *forget gate* f_t selects information to be forgotten, given the input vector x_t and previous hidden state h_{t-1} . To this effect a sigmoid layer σ (i.e. mapping states to values between 0 to forget, and 1 to keep) is applied to this concatenated input and the result multiplied with the cell state c_t , with weight matrices ω , v :

$$f_t = \sigma(\omega_f x_t + v_f h_{t-1} + b_f)$$

New information is passed through the *input gate* i_t , which selects the values to modify, with a tanh layer σ forcing the values to lie between -1 and 1. The product is then added to the cell state³:

$$\begin{aligned} i_t &= \sigma(\omega_i x_t + v_i h_{t-1} + b_i) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \varsigma(\omega_c x_t + v_c h_{t-1} + b_c) \end{aligned}$$

Finally, the *output gate* o_t selects the information from the cell state to be output as the hidden state. An element-wise tanh function is applied to the cell state values beforehand for normalisation:

$$\begin{aligned} o_t &= \sigma(\omega_o x_t + v_o h_{t-1} + b_{io}) \\ h_t &= o_t \circ \varsigma(c_t) \end{aligned}$$

³The operator \circ marks an element-wise product.

2. Theoretical Background

As with RNNs, the LSTM can be run in both directions concurrently, which is accordingly referred to as a bidirectional long short-term memory network (BiLSTM). As the LSTM structure only affects the internal makeup of the cell, outputs from both directionalities can be concatenated and used to the same effect as before.

The model presented in this thesis uses a BiLSTM network with a multi-layer FFNN as scoring function.

2.2.4. Vector Semantics & Embeddings

The central concept behind vector semantics can be traced back to Ludwig Wittgenstein, who claimed in his *Philosophical Investigations* (Wittgenstein, 2015):

Der Gebrauch des Wortes in der Sprache ist seine Bedeutung.

This general notion that a word is not the product of a logical definition, but shaped by its use, and as a result, context, was later formalised by Joos (1950), Harris (1954) and Firth (1957) as the *distributional hypothesis*. According to distributional semantics, a word can be defined through its distribution of the contexts it appears in.

In computational linguistics, this concept was later developed into a multitude of word embeddings, each aiming to quantify the contexts of a word through a mathematical vector representation. The simplest approach to calculate word vectors is a bag of words (BoW) method, where the other tokens within a certain window of every occurrence of a word are counted and put in a vector. Many n-gram models and measures like term frequency-inverse document frequency (tf-idf) (Salton and McGill, 1986), but also complex embeddings like Word2vec (Mikolov, Sutskever, et al., 2013; Mikolov, Chen, et al., 2013) and Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) are based on the BoW foundation.

This relationship is most apparent in Word2vec’s continuous bag of words (CBOW) model. The underlying FFNN aims to predict a word from context words within a certain window. The other included model, skip-gram, is trained on the opposite objective, predicting the context from a given word. The hidden states for each token are the true product of this model, as the average over the hidden states of each occurrence of the token in question.

Another popular approach are GloVe embeddings (Pennington et al., 2014), as they are close in performance to Word2vec embeddings trained on similar data and pre-trained embeddings are readily available. In contrast to Word2vec, GloVe aims to capture global count statistics, as well as local contexts. To this end, co-

2. Theoretical Background

occurrence matrices are employed, as was the case with earlier embedding methods like latent semantic analysis (LSA), which did however not produce the same semantic relationships that were possible to model using Word2vec (e.g. “man is to woman and king is to queen”). GloVe is trained on predicting the *ratio* in which two words co-occur. This follows the intuition that two similar words can be expected to occur in similar contexts, which lead to similar co-occurrence ratios. To put less importance on infrequent (i.e. noisy) words, and not to overemphasise frequent terms (e.g. *is*, *and*, *it*, etc.), the following weighting function is applied to updates:

$$w(x) = \min(1, (x/x_{max})^{\frac{3}{4}})$$

Pennington et al. use $x_{max} = 100$ as a frequency threshold, but this is obviously dependent on the corpus size.

Newer developments involve context-dependent embeddings, where the embedding of a word is also dependent on its context, which breaks with the dictionary-like structure of previous embeddings. Now the whole pre-trained model has to be employed, not just a table with tokens and corresponding embeddings. The system with the largest impact in this domain is the transformer-based language model presented in Devlin et al. (2019).

3. Related Work

3.1. Anaphora Resolution

Little work has been published on comparative anaphora resolution, specifically in the last ten years. There is however active research in related fields of anaphora resolution, such as coreference and bridging resolution. Methods shown to work on these tasks should, in theory, also perform reasonably well for comparative anaphora. In this Section previous machine learning systems and corpora for comparative anaphora resolution (3.1.1) are discussed, as well as neural recurrent approaches to coreference resolution (3.1.2) and work done in the field of bridging (3.1.3), which shares some overlap with the task at hand.

3.1.1. Comparative Anaphora Resolution

Modjeska (2003) presents a similar experimental setup to the one in this work, which is evaluated on two different data sets:

- a data set with all candidates preceding the actual antecedent removed,
- a more realistic data set, with all candidate antecedents in a two sentence window retained.

The data set is of similar size (500 samples) as the one presented in Section 4 and similarly pruned. List- and other-than-constructions were removed and only noun phrase (NP) antecedents considered. The samples of Modjeska’s corpus were drawn entirely from the Wall Street Journal (WSJ) documents from OntoNotes 5. The reported ratio of candidate to actual antecedents (500 : 2584) is about half of the 512 : 5690 split found in the data collect for this thesis. The systems tested in Modjeska (2003) did allow multiple (or no) candidates to be classified as antecedents and were evaluated using ten-fold cross-validation.

Crucially, Modjeska (2003) only considers “other”-anaphora, i.e. comparative anaphora which feature *other* or *another* as comparative modifiers. The system itself is largely based on Naïve Bayes and the semantic knowledge extraction from the web, as suggested in Modjeska et al. (2003). Candidate relations were fed into a search engine using the “X and other Y” list-construction, which is almost always

3. Related Work

indicative of an anaphoric relation (Modjeska, 2003). The number of yielded results for each pairing was used as a feature for the Naïve Bayes classifier.

3.1.2. Coreference Resolution

Two concepts used successfully to resolve coreference are of special importance, as they relate to this work: First, the idea of mention ranking, which lends itself naturally to a neural solution. And of course the RNN architecture, which allows each mention to be embedded within its own context for further classifying.

Neural networks, more specifically deep reinforcement learning, have been successfully employed for mention ranking for coreference resolution (Clark and Manning, 2016b; Clark and Manning, 2016a). Coreference through mention ranking compares mentions of a document pairwise and either selects them as coreferent, or not. Compared to partial cluster evaluation, this is a faster and more scalable approach, and as a result became standard in coreference resolution (Wiseman et al., 2015; Durrett and Klein, 2013). Clark and Manning (2016b) also use a concatenated semantic embedding and feature vector as input. They use reinforcement learning (Williams, 1992), which is not necessary for the comparative anaphora task, as the data provides a full supervision signal.

The end-to-end coreference resolution system presented in Lee, He, Lewis, et al. (2017) and Lee, He, and Zettlemoyer (2018) uses a BiLSTM to create a span representation, which in turn is fed into a scoring function. A very similar architecture has later been used for semantic role labelling (He et al., 2018), proofing an adaptability to different tasks. In contrast to this thesis, Lee, He, Lewis, et al. were also challenged with the task of selecting candidate spans. In their end-to-end coreference system, each span was represented by the first and last BiLSTM output, as well as an attention vector that models some form of “headiness”, i.e. what part of the span most constitutes its head, and a feature vector with domain and distance information. Each mention is paired with multiple other mentions, as well as a null-class, which is meant to be predicted if no mention is coreferent with the anaphor.

3.1.3. Bridging

Some data sets have been created for bridging resolution, most notably ISnotes (Markert, Hou, et al., 2012; Hou et al., 2013), which provides fine-grained annotations for information status and bridging relations for 50 documents from OntoNotes’ WSJ corpus. Because documents were selected for the bridging resolution task at large, ISnotes includes only about 100 samples of in-text comparative

3. Related Work

anaphora with nominal antecedents.

The ARRAU corpus (Poesio, Artstein, et al., 2008; Poesio, Grishina, et al., 2018; Uryupina et al., 2018) does provide a multitude of anaphoric annotated text, but unfortunately comparative anaphora is not marked individually, but only as part of bridging anaphora. The BASHI corpus Rösiger (2018) consists of further 459 bridging samples from the WSJ corpus, 70 of which can be classified as comparative anaphora. Most bridging research is contained to the ISnotes and ARRAU corpora, making OntoNotes’ WSJ corpus the de-facto benchmark for bridging resolution, as OntoNotes as a whole has become for coreference resolution.

Rösiger et al. (2018) include a neural relation classifier in their re-implemented rule-based systems for bridging (Hou et al., 2014) and coreference resolution (Björkelund and Kuhn, 2014). In Roesiger et al. (2018) the comparative anaphora task is explicitly excluded, as their system is not able to predict split antecedents, which frequently appear in list-constructions (e.g. “Libya, Sudan and other countries”).

3.1.4. Pre-Training

Only tangential work exists for the impact of pre-training on anaphora resolution tasks. One related task, where pre-training did have a positive impact, is Chinese zero pronoun resolution. Zero pronouns refer to gaps in sentences that are coreferent with an entity, which is providing context for understanding that gap. Liu et al. (2017) generate pseudo training data from randomly creating gaps for nouns and pronouns. This pre-training setup is highly similar to reading comprehension tasks, which is why the authors choose to employ a neural network model used previously on that task. Yin et al. (2018) are pre-training their zero pronoun resolution system on coreference data, but only go into little detail on the exact setup.

Pre-training is a common technique in reinforcement learning, which is why Clark and Manning (2016a) do employ some form of pre-training, but again do not go into much detail on the exact parameters or impact. For modern, learning intensive systems like Devlin et al. (2019), pre-training on an insurmountable amount of language data has become unavoidable to achieve good performance, which led to a rise of massive pre-trained models, which only need to be fine-tuned by the user.

4. Corpus

Two corpora were created specifically for this task, a manually annotated training corpus of comparative anaphora samples, and an automatically collected pre-training corpus from a pruned set of coreference mention pairs. The collection of the latter will be discussed in Section 4.1.1. Sections 4.1 and 4.2 deal with the preliminary filtering by anaphora and subsequent annotation of the samples.

All samples were collected from the OntoNotes 5 corpus and processed using the SpaCy dependency parser.

4.1. Preprocessing & Filtering

The OntoNotes corpus already provides sentence and word tokenisation. To allow parallel annotations, these, as well as any stop words or other fragments within the corpus were kept to that end.

The algorithm searches documents sentence-wise for occurrences of anaphora. Anaphora can generally be found through their unique modifiers:

- comparative adjectives, e.g. better, prettier;
- *more* or *less* in combination with an adjective, e.g. more important;
- a modifier from a closed class of adjectives, see M in Algorithm 1.

Because *than*-constructions generally appear more frequent than semantically encoded anaphora, sentences containing *than* were cut entirely. This may have led to some wrongful exclusions, but overall reduced annotation time considerably.

This preliminary filtering serves the main purpose of selecting documents for annotation. The algorithm therefore is build around recall, as to not exclude some class of anaphora by strict constraints. Notably, list-constructions had to be excluded manually. A condition for their exclusion could be included in a future implementation of this filter.

4. Corpus

```
 $M \leftarrow \{other, another, similar, further, separate, comparable, \\ added, supplemental, different, additional, extra, supplementary\}$   
if  $\neg \exists T : T.text = than \wedge T \in S.tokens$  then  
  for  $C \in S.noun\_chunks$  do  
    if  $C.head.text = others$  then  
      return  $C$   
    else  
      for  $T \in C.tokens$  do  
        if  $T.tag = JJR$  then  
          return  $C$   
        else if  $T.head.text = more \wedge T.tag = JJ$  then  
          return  $C$   
        else if  $T.head.text = less \wedge T.tag = JJ$  then  
          return  $C$   
        else if  $T.text \in M$  then  
          return  $C$   
        else  
          return  $\emptyset$   
      else  
        return  $\emptyset$ 
```

Algorithm 1: Check sentence S for comparative anaphora. Returns noun chunk C containing the anaphor if found, else *None*. M is a set of comparative modifiers not tagged as *JJR*.

As running a dependency parse over the whole corpus is computationally expensive, samples for annotation were collected one at a time, reducing the time needed for preprocessing after changing the filtering algorithm, as well as the startup time for the annotation tool, to a minimum. As a result, it is not possible to conclusively say how many possible cases for annotation there are in OntoNotes 5, given the algorithm presented above.

The head of the selected noun chunk is labelled as the anaphor head, while the heads of all other noun chunks and named entities are marked as candidate heads. The spans of chunks and named entities (NEs) are retained as a feature, as well as SpaCy’s part-of-speech (POS) annotation.

4.1.1. Pre-Training

The similarity between coreference and comparative anaphora resolution should, in theory, share some common indicators of anaphoric relations that can be pre-trained on a larger coreference data set and fine-tuned to the semantic relation

4. Corpus

between comparative antecedent and anaphora.

Coreference annotation was lifted from the CoNLL shared tasks on coreference (Pradhan, Ramshaw, et al., 2011; Pradhan, Moschitti, et al., 2012) on the OntoNotes data set (Pradhan, Hovy, et al., 2007). No document from the training set is included in pre-training, and no pre-training document in training. This is done prohibitively to not leak information.

Coreference mention spans are checked against noun chunks and named entities from the SpaCy parser. For the pre-training data set, all coreferent mention pairs with the antecedent immediately preceding a noun anaphor were considered. Further constraints were also considered, but would have led to a considerable drop in pre-training size. A more lenient span matching between noun chunks and coreference annotation could also have led to more data, but was deemed too prone to incorrect annotations.

The data set is created completely without human annotators, so the true quality could only be measured by random samples and the overall model’s performance when fed the new data.

4.2. Annotation

A custom annotation framework was built to provide a rapid annotation for the task. Since only the head of the closest anaphoric candidate has to be annotated, and candidates are already pre-selected through preprocessing, it suffices to number the available candidates and communicate the corresponding number of the true antecedent back to the system. If the true antecedent can not be found, the anaphor is idiomatic or syntactic, or the sample is corrupted in some other way, the sample can easily be discarded.

4.3. Analysis

As can be gathered from Table 2, the data for pre-training is slightly harder to classify, as there are more candidates from which to choose, as well as more distance between true antecedent and anaphor, both in terms of tokens and other candidates.

4. Corpus

	Train	Pre-Train
Number of samples	512	3825
Mean sample length	45.1	50.2
Mean number of candidates	11.1	12.8
Mean distance in tokens	14.2	20.1
Mean candidate distance	3.7	5.5

Table 2: Quantitative statistics of *Train* and *Pre-Train* data sets. Mean candidate distance measures the number of candidates between (and including) the true antecedent and the anaphor.

The removal of documents already covered in the *Train* data mainly affects the New Testament (NT) corpus, removing 165 documents. This is surprising when contrasted with the WSJ corpus, where only 106 documents were removed from pre-training, despite having the larger share of the training set. Since NT documents tend to be much shorter than those of other corpora (see Section 6.5.4), this omission might drive up the overall sample length and skew the statistics slightly. However, both in *Train* and *Pre-Train*, the NT corpus makes up about 30% of the data set, with the largest margin lying between the WSJ proportions of 35% and 25% respectively, as can be seen in Figure 5

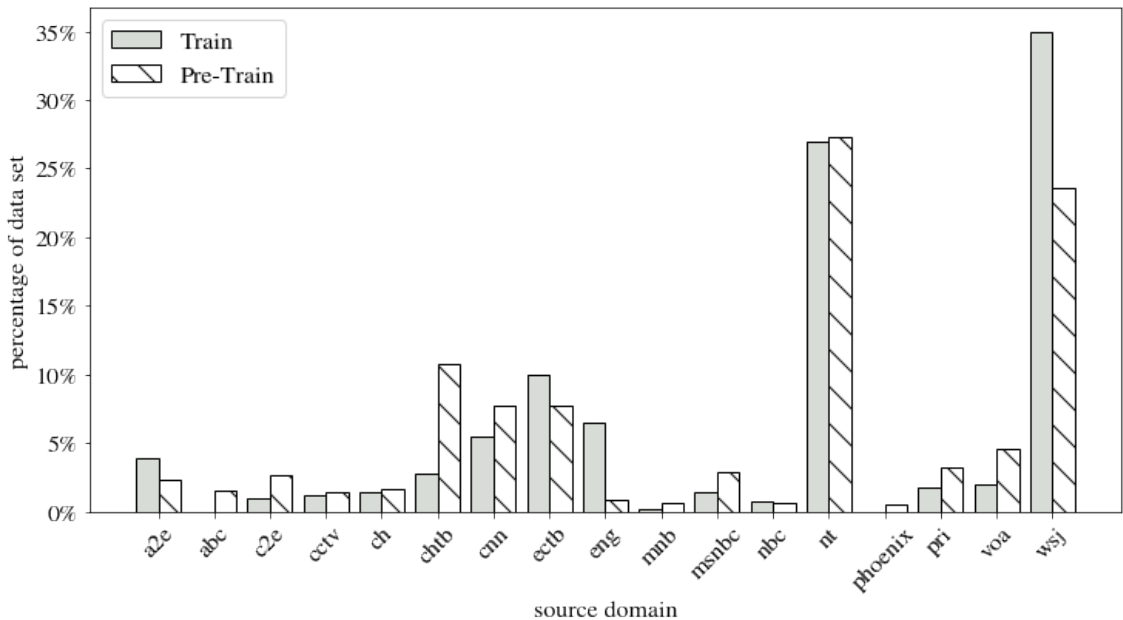


Figure 5: Proportion of source domain in both data sets. The Wall Street Journal (wsj) is skewed toward *Train*, the Chinese Treebank (ctb) towards *Pre-Train*.

5. Model

This work uses a RNN based on the LSTM (Hochreiter and Schmidhuber, 1997) architecture detailed in Section 2.2.3. From each cell’s hidden state an antecedent score is calculated using a FFNN. The logarithmic normalised exponential function (softmax) over all FFNN outputs gives the position of the prospective antecedent head. A diagram of the full architecture is shown in figure 6 below.

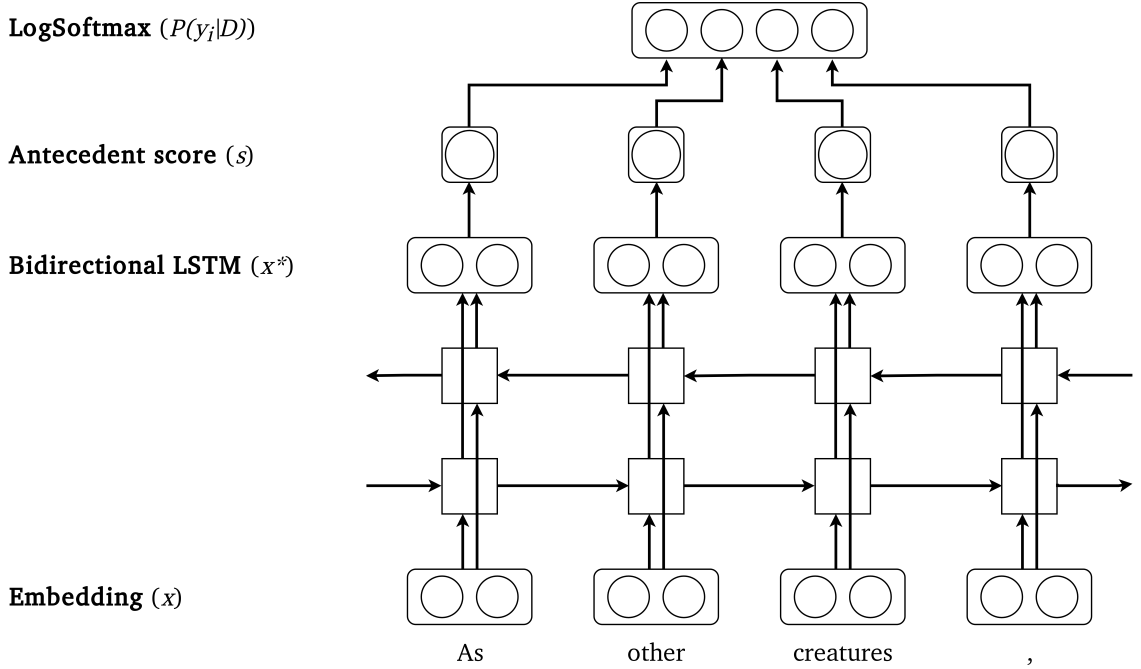


Figure 6: A diagram of the model’s layers. Each token is represented by the concatenation of their GloVe embedding and a syntactic feature vector. These are processed by a BiLSTM layer, and the concatenated cell outputs fed into a FFNN, which calculates an antecedent score. The model returns a softmax over all antecedent scores.

5.1. Input & Output

300 dimensional pre-trained GloVe embeddings trained on 6 billion tokens from Wikipedia are used as word vector representations. Pre-trained embeddings of size 50 were also briefly tested, but led to marginally worse results and were dropped as a result.

5. Model

	...	large	extent	that	they	won	the	other	party	...
x^{vocab} :	[513,	8329,	223,	2078,	121,	33,	839,	4943]
x^{head} :	[0,	1,	0,	1,	0,	0,	0,	3]
x^{POS} :	[JJ,	NN,	IN,	PRP,	VBD,	DT,	JJ,	NN]
x^{span} :	[6,	6,	0,	7,	0,	8,	8,	8]
x^{dist} :	[0,	$\frac{1}{2}$,	0,	1,	0,	0,	0,	0]
y :	[0,	0,	0,	1,	0,	0,	0,	0]

Table 3: Example of input and output vectors for a given sentence. Head, POS tag and span are randomly embedded into a vector space, words are embedded according to their GloVe vector and distance is given to the system as-is.

Mentions, including their heads and spans, were generated with a proprietary parser (Honnibal and Johnson, 2015). Mention head values correspond to *no label* (0), *is-candidate-head* (1) and *is-anaphora-head* (2). Because cataphoric constructions are quite rare, all candidates following the anaphora were given no label during testing to discourage such classifications. The POS tagger used the Penn Treebank tag set from OntoNotes 5 (Weischedel et al., 2013).

Wrong spans or heads were not pruned, but if the antecedent head was not found, a gold annotation was added by hand and marked as such. A neural network could pick up on the fact, that these candidates were hand annotated⁴, so only the correct antecedent for these samples is given during training. Cases, where the anaphora head was not correctly identified in preprocessing were excluded.

It may also be noted, that for pre-training on a coreference corpus, all found mention heads were given to the system as-is and were not manually annotated beyond the original coreference labels lifted from OntoNotes (Pradhan, Ramshaw, et al., 2011; Pradhan, Moschitti, et al., 2012).

The distance of a given candidate to its anaphor is determined by calculating the multiplicative inverse of the number of other candidate heads between the candidate and the following anaphor. Distance is only calculated for candidate heads, all other indices are given a distance of zero. This can be more thoroughly defined as the distance measure d_i at index i with anaphor head at index i_a and set of candidate heads C_a for anaphor a at index i_a :

$$d_i = \begin{cases} |\{i_c | c \in C_a; i \leq i_c < i_a\}|^{-1} & \text{if candidate head at index } i, \\ 0 & \text{else.} \end{cases}$$

⁴They could be syntactically distinct from some class of candidates the parser could identify, which, the more information is fed into a system, makes them uniquely identifiable as the correct candidate.

An example of the feature representations, as they appear before embedding, is given in Table 3.

5.2. Embedding Layer

Each token is fed into the LSTM layer as a concatenation of word embedding and a vector representation of the mention spans the token lies within, a distance measure to the following anaphora, as well as their POS tag and a one-hot-embedding of their head-property (or the absence thereof). Each feature is represented by a random projection into a multidimensional vector space, while words are represented by their GloVe vectors.

If a given token is part of multiple mentions (e.g. *U.S.* in *U.S. president*), all span embeddings of overlapping spans are summed. Since mention overlap occurs only on rare occasions, most mention representations for a given token are not transformed this way and the impact of transformations, when they occur, was not further explored.

All feature embeddings are concatenated and given to the BiLSTM cells as input.

5.3. BiLSTM Layer

This work uses a standard BiLSTM, described in more detail in Section 2.2.3. Because of the limited data size, I chose not to include bias or dropout in this layer. Likewise, I only use a single layer (i.e. no stacked LSTM).

Learning rate and hidden size were tuned before conducting the experiments. The performance of all tested parameters is outlined in Section 6.2.

5.4. Feed Forward Layer

The output of each BiLSTM cell is fed into a small FFNN which in turn calculates an *antecedent score* for each index. I use the same parameters for learning rate and hidden size for the two layers of our feed forward network. Again, because of the limited training size, a more complex network was not further explored. The logarithmic softmax is calculated over all antecedent scores to find the best antecedent candidate.

$$\varsigma(s)_i = \log\left(\frac{e^{s_i}}{\sum_{j=1}^K e^{s_j}}\right) \text{ for } i = 1, \dots, K \text{ and } s = (s_1, \dots, s_K) \in \mathbb{R}^K$$

5.5. Loss & Back-Propagation

As I use the logarithmic softmax as my final activation function, it is the obvious choice to use a negative log-likelihood loss, which worked well during testing:

$$L = \sum_{i=0}^{|\tilde{y}|} \begin{cases} -\log(\tilde{y}_i) & \text{if } i = i_c \\ -\log(1 - \tilde{y}_i) & \text{else.} \end{cases}$$

Since the system should only classify one index as correct, and only one index is correct, negative log-likelihood incentivises a confident prediction on one index, which falls in line with the softmax output. Back-propagation for this model mirrors the general LSTM back-propagation as outlined in Section 2.2.3 and is implemented using the PyTorch call stack algorithm.

If the mean loss over the last epoch does not drop after three epochs, the model at the lowest loss (*early stopping*) is returned for evaluation and, for pre-trained models, further trained on the main task.

6. Experiments & Results

6.1. Evaluation Measure

Because the softmax only allows exactly one antecedent and every sample features exactly one true antecedent, I measure each setup using micro- and macro-averaged success rate over a 10-fold cross-validation. For this system and this data set, as described above, this is equivalent to both true positive rate and positive predictive value. For a different activation function, these measures would have to be further differentiated.

Since samples that stem from the same document could lead to cross-contamination when split across multiple folds, each sample from one document is hashed into the same fold, which results in an uneven size of the train/test splits.

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.		
[37,	57,	63,	57,	52,	39,	31,	62,	52,	62]

Table 4: Size of each cross-validation fold. 5-fold cross-validation is performed on concatenations of two folds respectively.

For the pre-training experiments, all models are trained on the same 3825 samples from the *Pre-Train* corpus, but evaluated on the same ten folds as the other systems. The models trained for each fold are then kept, for the same folds, for the main training task.

Since the two-sentence window can feature multiple mentions of the same antecedent entity, the system only has to return any mention head of the true antecedent entity. Coreferent mentions are found using span annotations from the OntoNotes 5 coreference annotation (Weischedel et al., 2013). Since mention spans in our corpus were created by a dependency parser (Honnibal and Johnson, 2015) and OntoNotes was annotated by hand, some coreference relations might still be missing.

Statistically significant results ($p < 0.01$) are marked as such. Bootstrapping (Efron and Tibshirani, 1993) is used for significance testing. The exact method and further examinations of significance for this data set are explored in more detail in Section

6. Experiments & Results

6.6.

If not otherwise stated, any given report of success rate hereafter considers the micro-average result.

6.2. Parameter Tuning

Parameter tuning was performed on a five-fold cross-validation of the full 512 sample training set, with 3825 samples in the pre-training set. Because the limited test data leads to a high threshold for significant results, only parameter changes in octaves were considered. Changes in orders of magnitude would be nonsensical for almost every parameter for more than two or three variations (e.g. a batch size of 0.1 is not possible, while 1000 exceeds the amount of training data). A change in octave is still substantial, but allows a larger number of options to be tested.

Learning Rate	Epochs	Batch Size	Hidden Layer Size	Success Rate	
				Micro-Avg.	Macro-Avg.
2^{-11}	16	16	256	29.49%	28.68%
2^{-9}	"	"	"	29.10%	29.45%
2^{-7}	"	"	"	27.34%	27.06%
2^{-9}	4	16	256	27.54%	27.44%
"	8	"	"	28.32%	27.99%
"	32	"	"	30.47%	29.84%
"	64	"	"	28.32%	27.93%
2^{-9}	32	4	256	23.24% [‡]	23.44%
"	"	8	"	29.30%	29.43%
"	"	32	"	29.29%	29.37%
2^{-9}	32	16	64	24.80% [‡]	24.94%
"	"	"	128	30.27%	30.16%
2^{-9}	32	16	512	30.47%	30.31%
"	"	"	1024	28.32%	28.35%

Table 5: Parameters for further training were selected based on the results of a single run over a five fold cross-validation setup. Pre-Training was initialised with a learning rate of 2^{-10} , 4 epochs and a batch size of 32.

The parameters for pre-training were set beforehand somewhat arbitrarily and optimised in limited experiments after determining the optimal system parameters (see table 5). It should be noted, that only the marked setups ([‡]) perform significantly ($p < 0.01$) worse than the “best” setup (bold), which was used in all following experiments. These experiments seem to give lower limits of parameters to achieve a success rate of about 29%, but do not conclusively indicate an optimal setup above these minimal parameters.

6.3. Baselines

The architecture outlined in section 5 is evaluated against three naïve baselines. More complex baselines (e.g. greatest head embedding similarity) were discarded early because of poor performance and not further developed. Every baseline only considers preceding candidates, since cataphoric recency would lead more often than not to false positives. The performance of the hybrid system presented in Modjeska (2003) (54.25%) on a similar task, should be recognised at this point, but because of differences in data sets and the amount of preprocessing and external information used, can not be directly compared.

6.3.1. Random

The **random** baseline selects any preceding mention head to the anaphor. Because of the exclusion of cataphors, it is not a true random baseline, which would also consider heads of mentions that appear after the anaphor.

6.3.2. Recency

The **recency** baseline selects the closest preceding mention head and could be considered a majority baseline on the distance measure, as the plurality of anaphoric mentions precede the anaphora directly (see Section 4).

6.3.3. Head Match

The **head match** baseline selects the closest preceding mention with the same (lemmatised) head as the anaphora. If no mention features the same head within the two-sentence window, the recency baseline is used. This baseline suffers the most from incomplete entity matching, as head matches occur infrequently on the first preceding mention of an entity, but only directly preceding mentions are annotated for this task. The matched head would have to be marked as correct through the coreference matching process.

6.4. Results

The model is tested in three different permutations: The *Pre-Train* model is only trained on the filtered coreference samples. The *Train* model is trained exclusively on comparative anaphora samples. The *Full* model is trained on both, in succession.

Running the system with the parameters found most suitable in Section 6.2, the models yields the following results:

6. Experiments & Results

Full (1)	Train (2)	Pre-Train (3)	Recency (4)	Head Match (5)	Random (6)
<i>Macro-Average</i>					
32.75%	29.44%	25.67%	23.02%	21.92%	4.33%
<i>Micro-Average</i>					
32.81% [‡]	29.69% [†]	25.98%*	22.85%*	22.92%*	4.30%

Table 6: Macro- and micro-averaged results of all systems (BiLSTM with pre-training (1), BiLSTM without pre-training (2), BiLSTM only trained on coreference samples (3), recency, head match and random baselines (4-6)) over a ten-fold cross-validation on 512 data samples. Significance is highlighted for micro-averaged results⁵.

6.4.1. Ablation

The impact of individual covariants on the success rate was tested using the *Train* setup. The full list of tested permutations is given in Table 7.

Covariant	Success Rate		<i>p</i> -value
	Micro-Avg.	Macro-Avg.	
w/o POS	29.88%	29.57%	0.334
w/o Embeddings	27.15%	27.35%	0.231
w/o Heads	27.93%	28.26%	0.331
w/o Distance	33.59%	33.50%	0.004 [†]
w/o Spans	33.59%	33.88%	0.009 [†]
w/o Spans/Heads	24.80%	24.63%	0.022
w/o Spans/Distance	34.18%	33.89%	0.005 [†]
w/o Spans/Heads/Distance	26.95%	27.30%	0.159
only Embeddings	22.27%	22.15%	0.001 [↓]
only POS	25.00%	24.86%	0.041

Table 7: Micro- and macro-averaged success rates for multiple ablation experiments. Significance is tested against the results from the *Train* model.

H₀: There is no performance change from dropping a given covariant.

↑ indicates a significant increase, ↓ a significant decrease ($p < 0.01$).

Most results were not significant enough to warrant any conclusion, but distance and span annotation did impact performance negatively. The way the mention spans are conveyed to the system could lead to the incorrect predictions. Because each sample mention is labelled from left to right, the span IDs could be interpreted

⁵ ‡ : $p < 0.01$ for (3)-(6); † : $p < 0.01$ for (4)-(6); * : $p < 0.01$ for (6)

6. Experiments & Results

by the system as a very inconsistent distance measure. As the true antecedent tends to stay close to the anaphor, and size of samples varies by domain, span ID taken as distance could be a hindrance for transferring knowledge from one domain to another.

The systems using distance as a covariant classifies indices as antecedent heads two tokens farther from the anaphor head than their counterparts without distance features (14.2 vs. 12.3 tokens). As a consequence, the distance models may have learned to classify close to the mean distance (13.75 tokens), not close to the median (10 tokens). The latter could have yielded better results due to the long-tailed distribution of true antecedents at distance (see Figure 13). Only considering incorrect predictions, the mean distance to the anaphor head further increases to 15.7 and 13.6 tokens respectively.

The results after dropping distance and span labels can be found in Table 8 below.

Full (1)	Train (2)	Pre-Train (3)	Recency (4)	Head Match (5)	Random (6)
<i>Macro-Average</i>					
33.22%	32.54%	26.88%	23.02%	21.92%	4.33%
<i>Micro-Average</i>					
33.59% [‡]	32.42% [†]	27.15% [*]	22.85% [*]	22.92% [*]	4.30%

Table 8: Success rates for all systems after dropping distance measure and span labels. Significance is highlighted for micro-averaged results, as before⁶.

The *Full* model’s success rate does not see a significant increase from dropping span labels and distance. The *Pre-Train* model gains about 1%, which could be due to normal variation, while the *Train* model increases to the *Full* system’s level.

6.4.2. Training Data

To measure the impact of the provided training data (and judge whether more annotation would lead to a significant performance increase), I perform a 10-fold cross-validation with randomly sampled training sets of sizes 2^0 , 2^1 , and so forth, up to 449 (the training size for the largest test fold) from each training subset. For the *Full* model, each fold of each subset continues training on a copy of the same model pre-trained on 3825 coreference samples.

⁶ [‡] : $p < 0.01$ for (3)-(6); [†] : $p < 0.01$ for (4)-(6); ^{*} : $p < 0.01$ for (6)

6. Experiments & Results

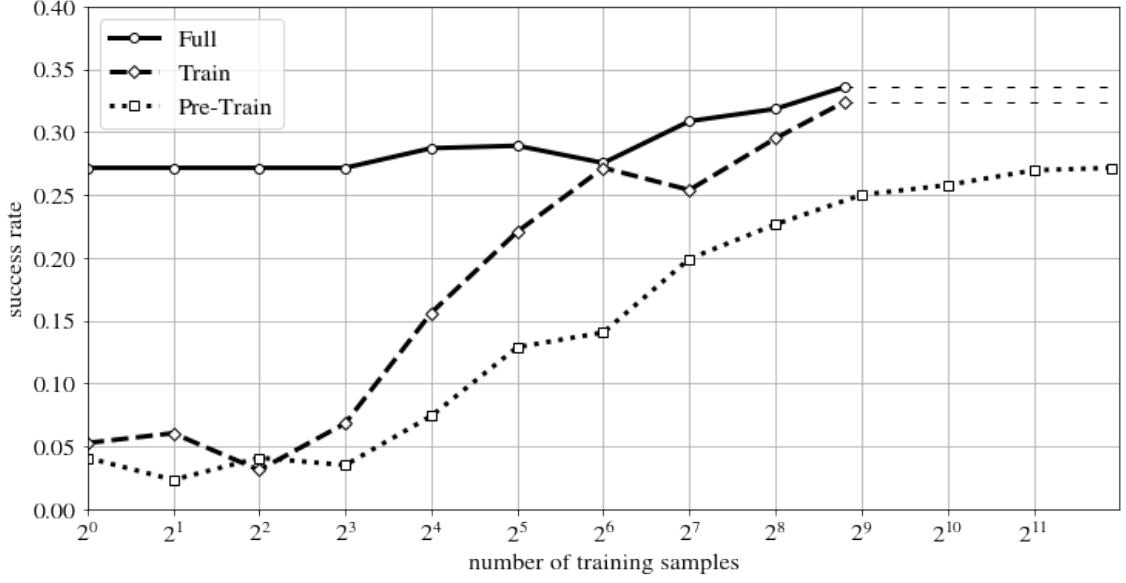


Figure 7: Success rate for size of training data. The *Full* system continues training from the rightmost *Pre-Train* model.

All models show a general upward trend, suggesting that more training data could lead to further increases. The head start from pre-training could not be retained for the *Full* system, with the success rate dropping to *Train* levels with only 64 samples.

For each run of any of the neural models, a major variation in performance could be observed, usually varying between 10 and 40 percent success rate between folds. As a consequence, more annotation, specifically for a validation set, could result in a noticeably higher success rate. This is even more so the case for pre-training, as a validation set could offer a condition for early stopping that is actually founded in the task, not in an approximation.

6.5. Error Analysis

6.5.1. Overlap

Figure 8 shows the overlap of predicted indices between each system (e.g. the *Full* and *Train* models predicted the same index as antecedent head 147 times when *Pre-Train* chose a different index). The gradual alignment of *Train* and *Full* clearly stems from a higher similarity in predictions. This could be due to more training “overwriting” previously learned knowledge from pre-training.

A lower learning rate could lead to more retainment of pre-trained knowledge. An alternative would be to feed coreference samples during further training or merging the data sets altogether.

The assumption that actual comparative anaphora data is to be of so much higher quality for training, as to be kept separate and trained on in succession, may have to be reevaluated, with both systems achieving rather similar results.

6.5.2. Non-Head Predictions

There is no hard constraint preventing the model from predicting tokens as antecedent heads, that have not been annotated as a candidate. There are a few cases of this being the correct annotation in training, as heads after the anaphor are not tagged. Nonetheless, both the *Train*, as well as the *Pre-Train* model learn to annotate only candidates eventually, as is evident in Figure 9. While non-head predictions are not common in the final systems, the *Pre-Train* model in particular needs many iterations to learn this constraint, despite not encountering any cataphoric samples during training.

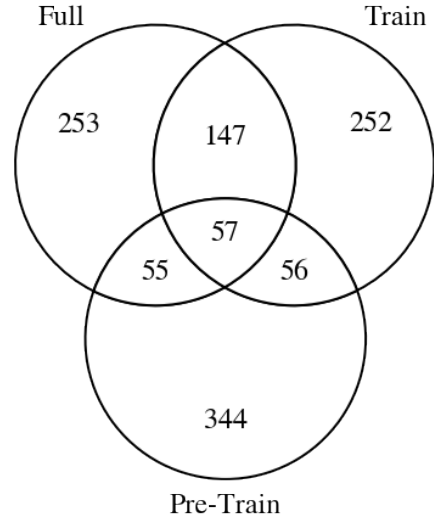


Figure 8: Overlap in predicted indices between the three models.

6. Experiments & Results

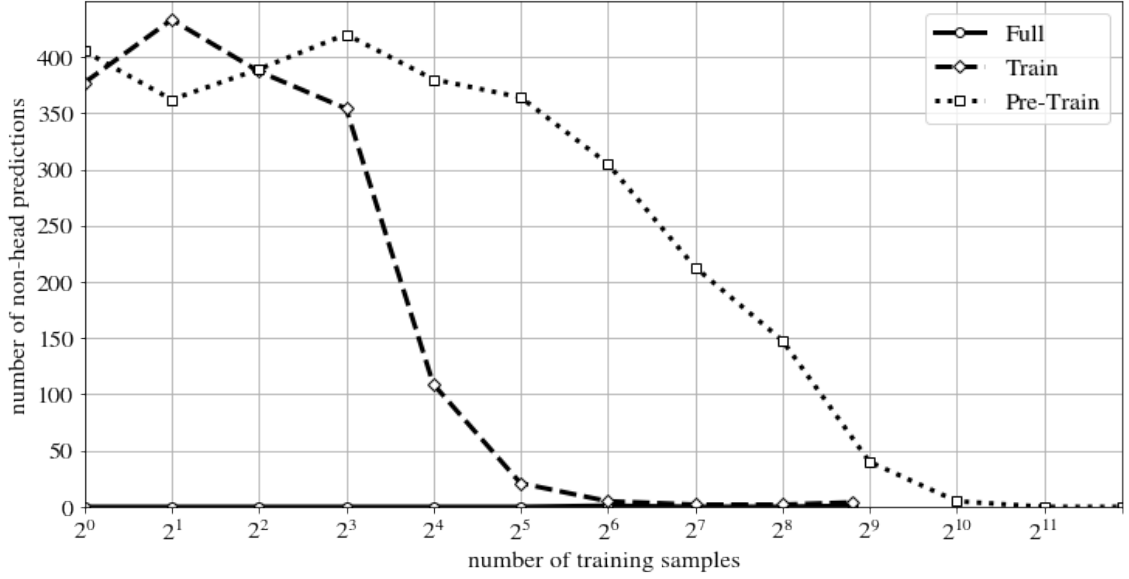


Figure 9: Number of non-head predictions, i.e. predictions on indices that have not been annotated as candidates, for each system and size of training set.

6.5.3. Part-of-Speech

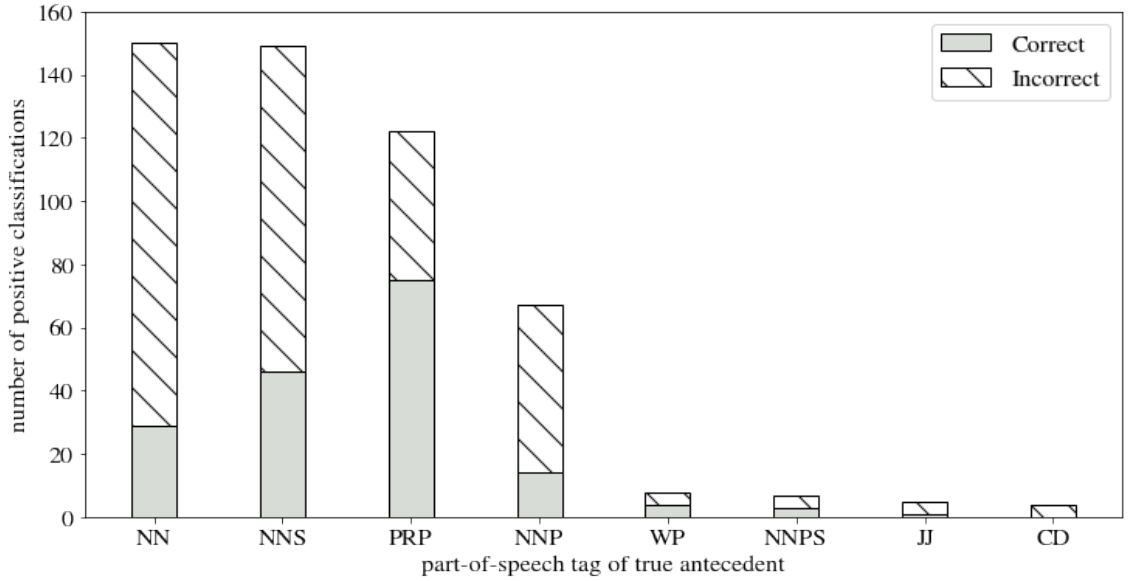


Figure 10: Number of positive classifications broken into correct and incorrect predictions, sorted by POS tag. The results are taken from the *Full* model's output.

The POS tag of the true antecedent could influence the system's decision, as some parts of speech might be more informative by nature for any given anaphora. Antecedents with proper nouns, singular as head performed much better than any other metric, while nouns, both singular and plural, achieved results slightly below

6. Experiments & Results

average. Proper nouns, singular fell in line with 28, %.

Other parts of speech, with 24 occurrences total, appeared too infrequent to make a qualitative statement. Of these other samples, four are non-nominal named entities, which were all adjectives indicating nationality, one of which can be seen in the following example from the corpus:

I believe Mr. Kageyama left out one major aspect of Japanese culture that permeated his piece: the belief in the superiority of *Japanese* culture and behaviour versus *others*.

The full breakdown of each POS tag can be seen in Figure 10. It might be a reasonable assumption that those POS tags, which only appear infrequently as antecedent head, might be harder for the model to classify as anaphoric. This is clearly the case for prepositions, which have the highest proportion of head appearances and by far the highest success rate. This is also mirrored for nouns, plural, which appear slightly more often as heads than their singular counterparts, and perform better accordingly.

6.5.4. Source Domain

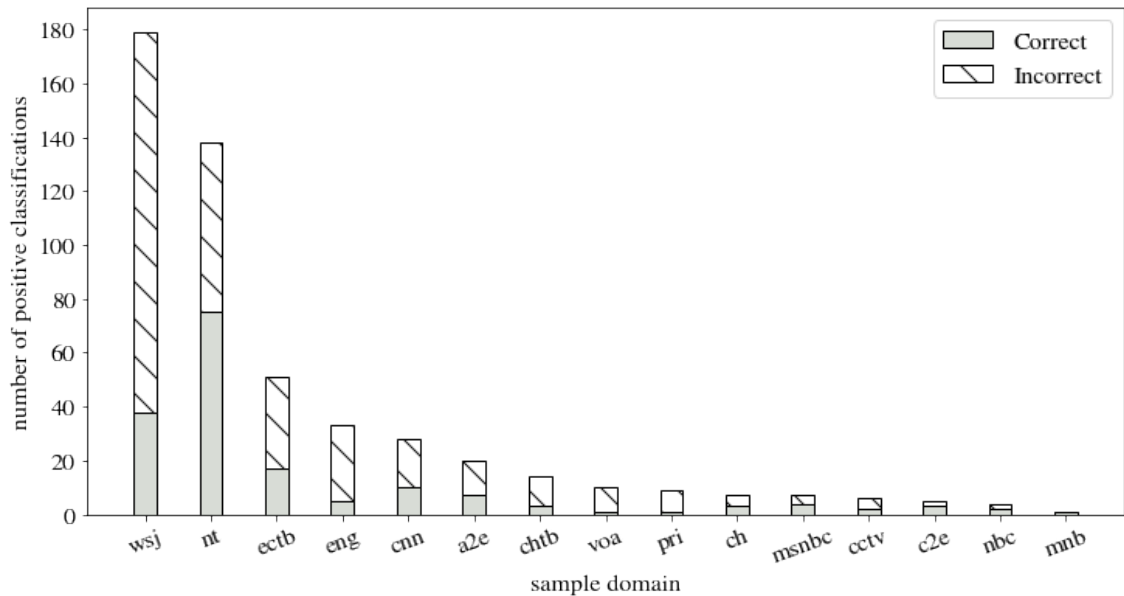


Figure 11: Number of positive classifications broken into correct and incorrect predictions, sorted by the domain of the sample. The results are taken from the *Full* model’s output.

As is evident from Figure 11, samples from the NT corpus were classified about half the time correctly, a major jump from the 21% achieved on the WSJ corpus. This may most obviously be due to the fact that the NT samples with a mean word

6. Experiments & Results

count of 28 are only half as long as those from the WSJ or English-Chinese Parallel Treebank (ECTB) corpora, which measure 53 words in the mean each.

6.5.5. Comparative Modifier

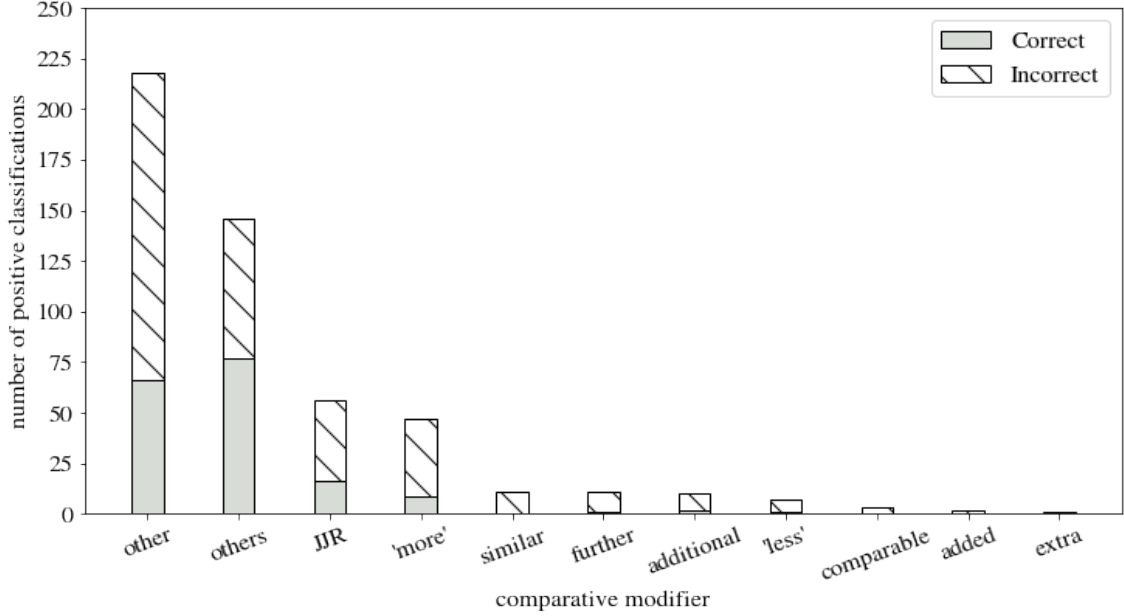


Figure 12: Number of positive classifications broken into correct and incorrect predictions, sorted by comparative modifier. The results are taken from the *Full* model’s output.

Figure 12 shows the success rate broken down for comparative modifiers. As expected, *other* appears most often in our data set, closely followed by *others*, which is largely because of its prevalence in the NT corpus. Morphological (e.g. “greater”, JJR) and syntactic comparatives (“*more* beautiful”) combined make up about a fifth of the data set. *Others*-anaphora are successfully resolved 48% of the time, *other*-anaphora coming in second with 28% success rate.

The relationship between the NT corpus and the comparative modifier *others* should be highlighted at this point. 53% of the occurrences of *others* appear in the NT samples, for the WSJ this percentage is only at 16%. As a result, 54% of the NT data set have *others* as the anaphor. It is unclear, if the high success rate for anaphors with *others* as comparative modifier is so high, because it so often appears in simple NT samples, or the NT samples are so easy to classify because they can easily learn from the many similar samples also employing *other* as modifier.

6. Experiments & Results

6.5.6. Distance

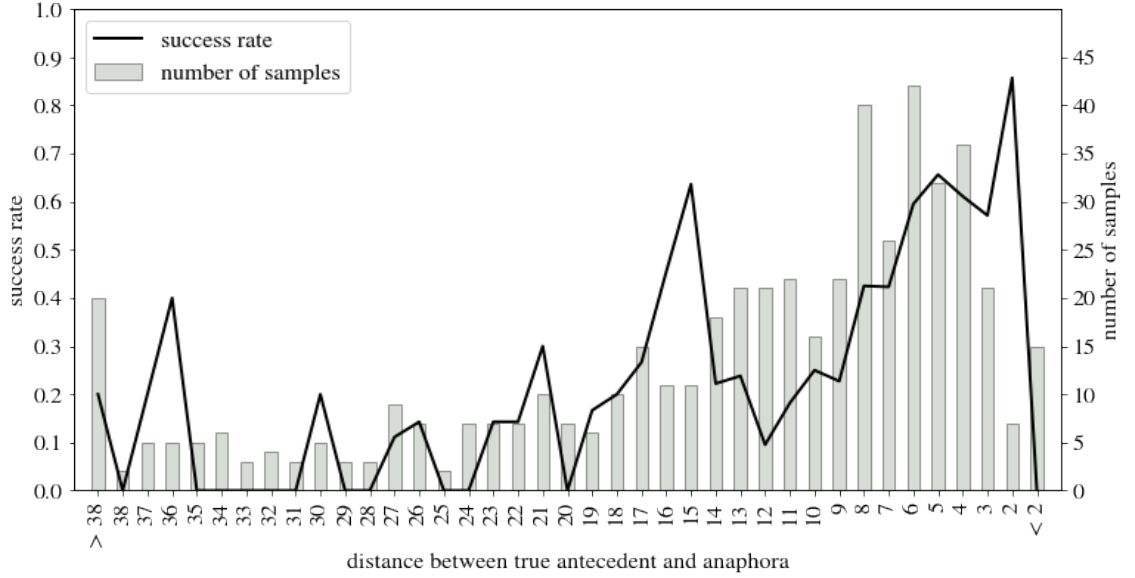


Figure 13: Success rate and number of samples at distance x between anaphora and antecedent head. The results are taken from the *Full* model’s output. The 15 samples with a distance of < 2 are all cataphoric.

Figure 13 shows the distribution of true antecedents at distance from the anaphora, as well as the success rate at each. While the first antecedent is the most frequent true antecedent, the average antecedent is in fourth place, due to the long-tailed nature of the distribution. Performance increases the closer the antecedent is to the anaphora. Whether this is because the model learned the majority class or because information of the anaphor is getting lost through the LSTM for farther antecedents is up for debate.

The *Train* model does classify more frequently at a higher distance than the *Full* model, with 40% more samples at a distance of over 20 tokens from the anaphora. On average, both systems are off from the true antecedent by 13.5 tokens, not counting correct predictions.

6.6. Significance

Since the underlying distribution of our system is not known, we use a non-parametric pairwise bootstrapping test for significance testing (Berg-Kirkpatrick et al., 2012). We sample 10^6 pseudo test sets from the merged cross-validation results of a 10-fold run of each variation of our system and count the number of

6. Experiments & Results

pseudo test sets that result in each success rate. The resulting distributions can be seen in Figure 14.

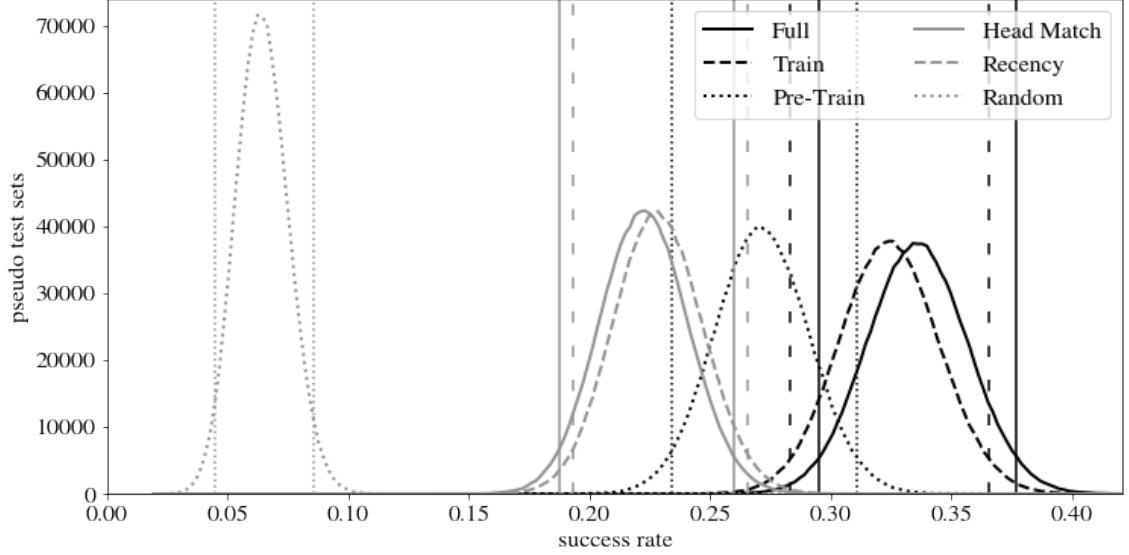


Figure 14: Bootstrapped confidence intervals for all systems with 10-fold cross-validation. The x-axis denotes success rate and the y-axis shows the number of bootstrapped test sets for which this result was achieved. Confidence interval ($C = 95\%$) are marked with vertical lines.

The confidence intervals in Figure 14 should not be confused for the p -values given in Table 6, which were calculated pairwise using the algorithm proposed in Berg-Kirkpatrick et al. (2012), as they show no correlation of specific systems for individual data points. For example, an arbitrary system A classifies samples $n \subset x$ correctly, while system B correctly predicts only samples $m \subset n$. As we cannot draw any precision tuples from this distribution, which would have system B giving the true and system A giving a false prediction, the bootstrapped p -value would equal zero. This does however have no impact on the overlap of the distribution of achieved accuracy over b samples, which informs the given confidence intervals (and also highlights an issue with calculating p -values from confidence intervals).

7. Conclusion

This thesis, at its core, investigates the relationship between different anaphora resolution tasks, and the approaches to each. This is evident by the experiments of pre-training a comparative anaphora system on the coreference task, and from employing a neural architecture previously successfully, albeit in a more complex whole, used for coreference resolution.

Section 4 introduces a new, cross-domain corpus for comparative anaphora resolution. With the exclusion of list- and than-constructions, it consists largely of non-trivial samples of in-text comparative anaphora, making it a tough benchmark for any comparative anaphora resolution system. This corpus is supported by a larger pre-train data set, gathered from coreferent mention pairs and filtered to suit the general makeup of the main task’s data set.

Section 5 presents a simple architecture meant to convey the semantic information found in word embeddings between anaphor and antecedent. The presented BiLSTM features a loss and output function fitted to the task. In addition to word embeddings and candidate annotation, distance, mention spans and POS tags are encoded to convey information intuitively relevant to the task. Although two of these features later prove to be detrimental to the performance, they give nonetheless insight into how representation shapes information in the case of neural networks.

Section 6 provides valuable insight for employing a recurrent network in the comparative anaphora task. The system only trained on selected coreference pairs does not fall far behind systems, proving a general close relation between the two tasks. Encoding additional features beyond word embeddings proved more difficult as expected. A distance measure, which is expected to perform well on this task, as it is a standard metric in almost every other anaphora resolution system (Modjeska, 2003; Lee, He, Lewis, et al., 2017; Roesiger et al., 2018), did lead to a performance decrease. The span label feature did also negatively affect the success rate, which can be more easily explained through its encoding, which could be misread as a (misleading) distance measure by a neural system.

Bible texts prove to be vastly different to the other domains in the data set. Not

7. Conclusion

only are the texts from the NT texts easier to classify due to their shorter length, but the vocabulary from which anaphor heads are drawn, is dominated by a single term, *others*, which is itself a kind of outlier for comparative anaphora, as it does not feature a comparative modifier.

Finally, this thesis lays further ground work for evaluating and annotating comparative anaphora. The data sets used are made freely available and present a tough to beat benchmark for any system to come. The recurrent model used in this thesis presents a simple neural baseline for future systems trained on this task.

8. Future Work

The first hypothesis stated in Section 1 could not be clearly answered. Even though every baseline was surpassed significantly, the results are below what would be expected on this task. Section 6.5 did address multiple possible causes for the low learning rate, but this inconclusive result does not clearly answer if the information relevant to the task is conveyed through word embeddings, or not. This work largely treats the LSTM as a black box, but the cell state should, if correctly trained, carry information, especially from the anaphor, to candidates and vice versa in the opposite direction. A closer analysis into how an LSTM may carry anaphor information should be looked into in future works on recurrent networks and anaphora resolution.

While this work showed that in certain limited training environments, pre-training certainly provides an advance that the plain training model has to catch up to. In the conducted experiments it eventually does.

As noted in 6.4.2, further annotation might lead to a significant increase in performance, provided the trends indicated on the limited experiments conducted in this work can be generalised⁷. The process in which coreference samples are collected could be expanded, as the gradient descent works much smoother on the *Pre-Train* set, as on *Train*. A larger pool of training samples would also allow the creation of a validation set, making a better condition of early stopping possible. This is especially useful for pre-training, as early stopping is so far conditioned on the approximated coreference task.

A transformer architecture could also be considered, as it performed well in other NLP tasks Devlin et al. (2019). Training a neural network as outlined in Vaswani et al. (2017) is however highly resource expensive and the necessity and value of using such a system on this task should be determined before conducting research in this direction.

⁷Propædeutics for further annotation in accordance with the data used to educate our system are provided among the appendices of this thesis.

Bibliography

- Baumann, Stefan (2012). “Referential and lexical givenness: Semantic, prosodic and cognitive aspects Stefan Baumann, Arndt Riester”. In: *Prosody and meaning* 25, p. 119.
- Berg-Kirkpatrick, Taylor, David Burkett, and Dan Klein (2012). “An Empirical Investigation of Statistical Significance in NLP”. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 995–1005.
- Björkelund, Anders and Jonas Kuhn (2014). “Learning structured perceptrons for coreference resolution with latent antecedents and non-local features”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47–57.
- Carter, David (1987). *Interpreting anaphors in natural language texts*. Halsted Press.
- Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734.
- Clark, Kevin and Christopher D Manning (2016a). “Deep Reinforcement Learning for Mention-Ranking Coreference Models”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2256–2262.
- Clark, Kevin and Christopher D Manning (2016b). “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)*, pp. 643–653.
- Deemter, Kees van and Rodger Kibble (2000). “On coreferring: Coreference in MUC and related annotation schemes”. In: *Computational linguistics* 26(4), pp. 629–637.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Bibliography

- Durrett, Greg and Dan Klein (2013). “Easy victories and uphill battles in coreference resolution”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1971–1982.
- Efron, Bradley and Robert J Tibshirani (1993). *An introduction to the bootstrap*. CRC press.
- Elman, Jeffrey L (1990). “Finding structure in time”. In: *Cognitive science* 14(2), pp. 179–211.
- Firth, J. (1957). “A Synopsis of Linguistic Theory 1930-1955”. In: *Studies in Linguistic Analysis*. reprinted in Palmer, F. (ed. 1968) *Selected Papers of J. R. Firth*, Longman, Harlow. Philological Society, Oxford.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.
- Harris, Zellig S (1954). “Distributional structure”. In: *Word* 10(2-3), pp. 146–162.
- He, Luheng, Kenton Lee, Omer Levy, and Luke Zettlemoyer (2018). “Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics: Melbourne, Australia, pp. 364–369. URL: <https://www.aclweb.org/anthology/P18-2058>.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9(8), pp. 1735–1780.
- Honnibal, Matthew and Mark Johnson (2015). “An Improved Non-monotonic Transition System for Dependency Parsing”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics: Lisbon, Portugal, pp. 1373–1378. URL: <https://aclweb.org/anthology/D/D15/D15-1162>.
- Hou, Yufang, Katja Markert, and Michael Strube (2013). “Global inference for bridging anaphora resolution”. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 907–917.
- Hou, Yufang, Katja Markert, and Michael Strube (2014). “A rule-based system for unrestricted bridging resolution: Recognizing bridging anaphora and finding links to antecedents”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2082–2093.
- Joos, Martin (1950). “Description of language design”. In: *The Journal of the Acoustical Society of America* 22(6), pp. 701–707.
- Lee, Kenton, Luheng He, Mike Lewis, and Luke Zettlemoyer (2017). “End-to-end Neural Coreference Resolution”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 188–197.

Bibliography

- Lee, Kenton, Luheng He, and Luke Zettlemoyer (2018). “Higher-Order Coreference Resolution with Coarse-to-Fine Inference”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 687–692.
- Liu, Ting, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang, and Guoping Hu (2017). “Generating and Exploiting Large-scale Pseudo Training Data for Zero Pronoun Resolution”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 102–111.
- Markert, Katja and Udo Hahn (2002). “Understanding metonymies in discourse”. In: *Artificial Intelligence* 135, pp. 145–198.
- Markert, Katja, Yufang Hou, and Michael Strube (2012). “Collective classification for fine-grained information status”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pp. 795–804.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781*.
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119.
- Modjeska, Natalia N, Katja Markert, and Malvina Nissim (2003). “Using the web in machine learning for other-anaphora resolution”. In: *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, pp. 176–183.
- Modjeska, Natalia Nygren (2003). “Resolving other-anaphora”. PhD thesis. University of Edinburgh.
- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: <http://www.aclweb.org/anthology/D14-1162>.
- Poesio, Massimo, Yulia Grishina, Varada Kolhatkar, Nafise Moosavi, Ina Roesiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, et al. (2018). “Anaphora resolution with the ARRAU corpus”. In: *Proceedings of the First Workshop on Computational Models of Reference, Anaphora and Coreference*, pp. 11–22.
- Poesio, Massimo, Ron Artstein, et al. (2008). “Anaphoric Annotation in the ARRAU Corpus.” In: *LREC*.
- Pradhan, Sameer S, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel (2007). “Ontonotes: A unified relational semantic rep-

Bibliography

- resentation”. In: *International Conference on Semantic Computing (ICSC 2007)*. IEEE, pp. 517–526.
- Pradhan, Sameer, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang (2012). “CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes”. In: *Joint Conference on EMNLP and CoNLL-Shared Task*, pp. 1–40.
- Pradhan, Sameer, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue (2011). “Conll-2011 shared task: Modeling unrestricted coreference in ontonotes”. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, pp. 1–27.
- Roesiger, Ina, Arndt Riester, and Jonas Kuhn (2018). “Bridging resolution: Task definition, corpus resources and rule-based experiments”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics: Santa Fe, New Mexico, USA, pp. 3516–3528. URL: <https://www.aclweb.org/anthology/C18-1298>.
- Rosenblatt, Frank (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65(6), p. 386.
- Rösiger, Ina (2018). “Bashi: A corpus of wall street journal articles annotated with bridging links”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.
- Rösiger, Ina, Maximilian Köper, Kim Anh Nguyen, and Sabine Schulte im Walde (2018). “Integrating predictions from neural-network relation classifiers into coreference and bridging resolution”. In: *Proceedings of the First Workshop on Computational Models of Reference, Anaphora and Coreference*, pp. 44–49.
- Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). “Learning representations by back-propagating errors”. In: *Nature* 323(6088), p. 533.
- Salton, Gerard and Michael J. McGill (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.: New York, NY, USA. ISBN: 0070544840.
- Uryupina, Olga, Ron Artstein, Antonella Bristot, Federica Cavicchio, Francesca Delogu, Kepa J Rodriguez, and Massimo Poesio (2018). “Annotating a broad range of anaphoric phenomena, in a variety of genres: The ARRAU corpus”. In: *Natural Language Engineering*, pp. 1–34.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is all you need”. In: *Advances in neural information processing systems*, pp. 5998–6008.
- Weischedel, Ralph, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et

Bibliography

- al. (2013). “Ontonotes release 5.0 LDC2013T19”. In: *Linguistic Data Consortium, Philadelphia, PA* 23.
- Williams, Ronald J (1992). “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8(3-4), pp. 229–256.
- Wiseman, Sam, Alexander M Rush, Stuart Shieber, and Jason Weston (2015). “Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1416–1426.
- Wittgenstein, Ludwig (2015). “Bergen Nachlass Edition”. In: *Wittgenstein Source (2009-)*. Ed. by the Wittgenstein Archives at the University of Bergen under the direction of Alois Pichler. URL: [http://reader.wittfind.cis.lmu.de/reader/Ms-114,120v\[5\]](http://reader.wittfind.cis.lmu.de/reader/Ms-114,120v[5]).
- Yin, Qingyu, Yu Zhang, Wei-Nan Zhang, Ting Liu, and William Yang Wang (2018). “Deep Reinforcement Learning for Chinese Zero Pronoun Resolution”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1, pp. 569–578.

A. Notes on Annotation

While annotating for this task I largely followed my own informed intuition. The following is a by no means complete list of cases, which I deemed non-anaphoric or which I thought complex enough to warrant further comment:

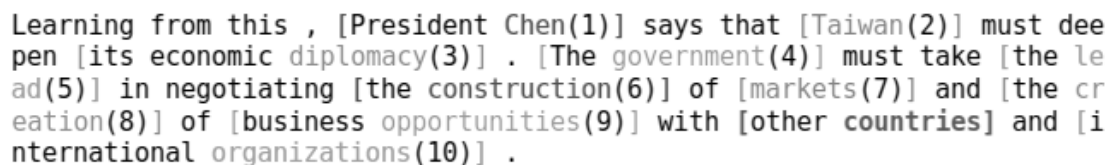
- “The older generation.” “The older brother.” If not clearly indicated otherwise, were considered idiomatic and discarded.
- “On the other hand/side.” Discourse markers, discarded.
- “More” without adjective. Largely exophoric, took up a lot of annotation time. Could be excluded to that end.
- Non-anaphoric lists. If the antecedent is not referenced in the surrounding list construction, the sample was annotated.
- Split antecedents. The closest member of the split antecedent to the anaphor head was annotated.
- Cataphora. Annotated, but not given as candidate during testing.
- “The more important question.” Discourse marker, discarded.
- “The other argument/thing.” Discourse marker, discarded.
- Elliptic constructions, zero pronouns. Difficult to find with parser, therefore not included.

B. Notes on Implementation

The full project was implemented in Python 3.7.3⁸. I used the following proprietary packages with the stated version below:

<i>Package</i>	<i>Version</i>		
bcolz	1.2.1	notebook	6.0.0
beatifulsoup4	4.7.1	numpy	1.16.4
conda	4.7.12	pandas	0.24.2
html5lib	1.0.1	scikit-learn	0.21.2
jupyter	1.0.0	scipy	1.3.0
matplotlib	3.1.0	spacy	2.2.1
nltk	3.4.4	tqdm	4.32.1
torch	1.2.0		

Most notably, the annotation tool (see Figure 15) relied heavily on SpaCy’s dependency parser to identify anaphora and antecedent candidates by their syntactic structure, their POS tags or the NE parse. This work could not have examined non-other-anaphora if the dependency and NE parse, and if host of other NLP features were not so conveniently bundled.



Learning from this , [President Chen(1)] says that [Taiwan(2)] must deepen [its economic diplomacy(3)] . [The government(4)] must take [the lead(5)] in negotiating [the construction(6)] of [markets(7)] and [the creation(8)] of [business opportunities(9)] with [other countries] and [international organizations(10)] .

Figure 15: Interface for annotation. The correct antecedent is marked by the annotator by ID. The head and span boundaries are highlighted.

The system itself was entirely written in PyTorch. This allowed for easy modularisation and quick changes for the many experimental setups this work went through.

Visualisations were done with Matplotlib.

⁸The full code and corpus will be made available at gitlab.cl.uni-heidelberg.de/zimmermann/bachelor.