Natural Language Processing in Adversarial Settings and Beyond: Benefits and Risks of Text Classification, Transformation, and Representation

Tommi Gröndahl
Natural Language Processing in Adversarial Settings and Beyond: Benefits and Risks of Text Classification, Transformation, and Representation

Tommi Gröndahl

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the room R001 of the school on 23 August 2021 at 12.00.

Aalto University
School of Science
Department of Computer Science
Secure Systems Group
Supervising professor
Professor N. Asokan, University of Waterloo, Canada / Adjunct Professor, Aalto University, Finland

Thesis advisor
Professor N. Asokan, University of Waterloo, Canada / Adjunct Professor, Aalto University, Finland

Preliminary examiners
Assistant Professor Valerio Baslie, University of Turin, Italy
Associate Professor Rachel Greenstadt, The New York University Tandon School of Engineering, United States

Opponent
Professor Jörg Tiedemann, University of Helsinki, Finland
Author
Tommi Gröndahl

Name of the doctoral dissertation
Natural Language Processing in Adversarial Settings and Beyond: Benefits and Risks of Text Classification, Transformation, and Representation

Publisher
School of Science

Unit
Department of Computer Science

Series
Aalto University publication series DOCTORAL DISSERTATIONS 94/2021

Field of research
Computer Science

Manuscript submitted 28 May 2021  Date of the defence 23 August 2021

Permission for public defence granted (date) 30 June 2021  Language English

☐ Monograph  ☑ Article dissertation  ☐ Essay dissertation

Abstract
Natural language processing (NLP) has developed significantly during recent years, with important consequences that extend beyond its immediate domain. The increased availability of NLP technologies has repercussions for information security and privacy in particular, both positive and negative. For example, classifying text based on semantic content or writing style has many benign uses, but also allows adversarial application for censorship or violations of privacy. Conversely, automatic text transformation can be used to perform model evasion attacks as well as defend against illegitimate profiling of text. This dissertation investigates the performance and security implications of NLP techniques across multiple tasks, with a focus on adversarial settings.

We first explored how well state-of-the-art text classification techniques can detect various types of adversarial text, such as deception or hate speech. Here, we observed that classifiers tend to get caught on simple features regardless of model architecture, which can make them unreliable and vulnerable to evasion. Instead of complicating the model alone, increasing the training dataset is needed for improving performance. We further demonstrated that text transformation can successfully be used to expand training data artificially. However, some adversarial text classes – such as deception – are likely too context-dependent to be reliably detected by available techniques.

We also applied text transformation to counteract classification, from both an attacker’s and a defender’s perspective. A major finding was that deep neural networks (DNNs) were unreliable at maintaining semantic content across transformations, in contrast to rule-based techniques that allow restrictive control of the output. On the other hand, DNNs are more flexible and can generate more variable texts than symbolic rules alone. This illustrates the complementary relationship between DNN-based and rule-based NLP, which speaks against discarding either. For mitigating model evasion, we show adversarial training to be beneficial against both kinds of techniques.

Across both text classification and transformation tasks, the importance of input data representation becomes apparent. This has broad relevance in a variety of NLP settings. Motivated by recent developments in linguistic theory, we show that effective semantic representations can be attained with far fewer semantic roles than in prior formalisms. Based on this, we present a novel format that permits easy but highly detailed information retrieval, as well as straight-forward integration with DNNs as vectorized input. In addition to demonstrating the format’s ability to retain information despite its structural simplicity, we applied it to parallel corpus extraction and text transformation tasks that resulted in multiple novel datasets we provide as open-access.

Keywords
- text classification
- text transformation
- text representation
- model evasion
- deception
- hate speech
- stylometry
- style transfer
- data augmentation
- semantics


ISSN (printed) 1799-4934  ISSN (pdf) 1799-4942

Location of publisher Helsinki  Location of printing Helsinki  Year 2021

Preface

This dissertation is the outcome of my doctoral studies at the Secure Systems Group in Aalto University. I would like to thank professor N. Asokan for welcoming me into the group, and having an open mind about expanding our focus without letting me lose touch with my main areas of interest. His supervision has been very useful in the development of my identity as a researcher, in both content and practice. My other co-authors were also invaluable for their collaboration and help: Mika Juuti, Luca Pajola, Mauro Conti, and Adrian Flanagan. I additionally want to thank Andrei Kazlouski and Sam Spilsbury for their important assistance and contributions. For funding, I am grateful to Helsinki Doctoral Education Network in Information and Communications Technology (HICT).

Beyond direct collaborators, I received much help from my co-workers at Aalto University who I had interesting discussions with about work and beyond: Hans Liljestrand, Andrew Paverd, Shohreh Hosseinzadeh, Samuel Marchal, Setareh Roshan, Lachlan Gunn, Sebastian Szyller, Buse Gul Atli, Verónica Toro, Thomas Nyman, Amit Tambe, Sanna Suoranta, professors Tuomas Aura and Janne Linqvist, and many others. Special thanks to Niina Idänheimo for being a great person to share an office with, and all the much-needed help with paperwork. My cognitive science colleagues have also been instrumental in my academic development. I especially want to thank Otto Lappi, Saara Huhmarniemi, Henri Kauhanen, Anna-Mari Rusanen, and everyone from our linguistics seminar in 2011–2016.

Finally, I would like to thank my family and friends: my parents Laura Gröndahl and Erkki Aura; and my group of friends, especially Heidi Sahlberg, Erno Paunonen, Viljami Venekoski, Iina Ala-Kurikka, and Aino Ruskomaa. Lastly, enormous thanks to Annemarie.

Espoo, July 8, 2021,

Tommi Gröndahl
Contents

Preface 1

Contents 3

List of Publications 5

Author’s contributions 7

List of Figures 9

List of Tables 11

1. Introduction 19
   1.1 Research questions .................................. 21
   1.2 Main results ........................................... 22
   1.3 Outline .................................................. 23

2. Text classification 25
   2.1 Features and classifier architectures .................... 25
      2.1.1 Features ............................................. 26
      2.1.2 Simple machine learning architectures ............. 28
      2.1.3 Deep neural networks ................................ 30
   2.2 Thematic content ........................................ 34
      2.2.1 Adversarial text: deception and hate speech ...... 35
      2.2.2 Author profiling by writing style ..................... 41
      2.2.3 Detailed classification based on semantic and grammatical structure .......................... 43
   2.3 Summary .................................................. 45

3. Automatic text transformation 47
   3.1 Text transformation techniques .......................... 47
      3.1.1 Rule-based text transformation techniques ...... 48
      3.1.2 Encoder-decoder techniques .......................... 51
List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s contributions

Publication I: “Text Analysis in Adversarial Settings: Does Deception Leave a Stylistic Trace?”

I carried out the survey and wrote the paper with feedback and assistance from my co-author.

Publication II: “All You Need is “Love”: Evading Hate Speech Detection”

The evasion attacks were designed by me and Mika Juuti, in an equal contribution. Implementation of the attacks was evenly divided between me and Luca Pajola, who ran the experiments. I wrote the paper with Mika Juuti, and the other authors provided additional help.

Publication III: “Effective Writing Style Transfer via Combinatorial Paraphrasing”

I designed and implemented ParChoice and the experiment settings. Andrei Kazlouski and Sam Spilsbury helped with running experiments in early stages of the project. I wrote the paper, with feedback and assistance from my co-author.

Publication IV: “A Little Goes a Long Way: Improving Toxic Language Classification Despite Data Scarcity”

Me and Mika Juuti took equal responsibility of designing and implementing the data augmentation techniques and experiments. Mika Juuti ran
the experiments apart from Section 4.5. Adrian Flanagan ran the ROC-AUC experiments and wrote Section 4.5. I and Mika Juuti led the writing of the paper, assisted by the other co-authors.


I designed EAT, designed and ran the experiments, and wrote the paper. Luca Pajola helped with implementation in early stages of the project.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Fully connected DNN with two hidden layers.</td>
<td>30</td>
</tr>
<tr>
<td>2.2</td>
<td>CNN with one convolution layer and one pooling layer.</td>
<td>31</td>
</tr>
<tr>
<td>2.3</td>
<td>Vanilla RNN</td>
<td>32</td>
</tr>
<tr>
<td>2.4</td>
<td>LSTM and GRU cells</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Encoder-decoder RNN</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>Transformer network</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>ParChoice pipeline</td>
<td>62</td>
</tr>
<tr>
<td>4.1</td>
<td>CBOW</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>Skip-gram.</td>
<td>76</td>
</tr>
<tr>
<td>4.3</td>
<td>EAT2seq and EAT-SimpleNLG pipelines</td>
<td>104</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Classification of publications in this dissertation.</td>
<td>22</td>
</tr>
<tr>
<td>2.1</td>
<td>Examples of the Writeprints features</td>
<td>28</td>
</tr>
<tr>
<td>2.2</td>
<td>The most common linguistic cues to deception</td>
<td>36</td>
</tr>
<tr>
<td>3.1</td>
<td>Data augmentation techniques used in Publication IV</td>
<td>57</td>
</tr>
<tr>
<td>4.1</td>
<td>Taxonomy between three prior semantic metalanguages used in NLP</td>
<td>74</td>
</tr>
<tr>
<td>4.2</td>
<td>Syntactic positions in the English clause</td>
<td>85</td>
</tr>
<tr>
<td>4.3</td>
<td>Grammatical features in EAT</td>
<td>98</td>
</tr>
<tr>
<td>4.4</td>
<td>Conversion between MRS$_{52}$, MRS-EAT$_9$, and MRS-EAT$_3$</td>
<td>101</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison between MRS$_{52}$, MRS-EAT$_9$, and MRS-EAT$_3$ in text reconstruction performance</td>
<td>102</td>
</tr>
<tr>
<td>4.6</td>
<td>Parallel corpora between grammatical classes</td>
<td>103</td>
</tr>
<tr>
<td>5.1</td>
<td>Classification of publications in this dissertation.</td>
<td>108</td>
</tr>
</tbody>
</table>
Abbreviations

<EOS>  end-of-sequence token.
<SOS>  start-of-sequence token.
[CLS]  initial token in BERT input sequences.

A^{4}NT  Author Attribute Anonymity by Adversarial Training of Neural Machine Translation.

AMR  Abstract Meaning Representation.

BLEU  Bilingual Evaluation Understudy.

BT  back-translation.

C  complementizer head (or any syntactic head in the C-domain).

CAE  cross-aligned autoencoder.

CBOW  contextual bag-of-words.

CNN  convolutional neural network.

Comp,XP  complement of the syntactic phrase XP in X’-theory.

CONN  connective.

CP  complementizer phrase.

DMRS  Dependency-MRS.

DNN  deep neural network.

DR  discourse referent.

DRS  discourse representation structure.

DRT  Discourse Representation Theory.
Abbreviations

**Ext, X**  external syntactic argument of X.

**G&B**  Government & Binding theory.

**GAN**  generative adversarial network.

**GloVe**  Global Vectors for Word Representation.

**GRU**  gated recurrent unit.

**HPSG**  Head-driven phrase structure grammar.

**ILT**  iterative language translation.

**Int, X**  internal syntactic argument of X.

**LF**  logical form.

**LM**  language model.

**LogLoss**  logistic loss = cross-entropy loss.

**LR**  logistic regression.

**LSTM**  long short-term memory network.

**METEOR**  Metric for Evaluation of Translation with Explicit Ordering.

**ML**  machine learning.

**MLP**  multilayer perceptron.

**MRS**  Minimal Recursion Semantics.

**MT**  machine translation.

**N**  noun.

**NLP**  natural language processing.

**NMT**  neural machine translation.

**NP**  noun phrase.

**P**  preposition.

**PCFG**  probabilistic context-free grammar.

**POS**  part-of-speech.

**PP**  prepositional phrase.
**PPDB** Paraphrase Database.

**Q** quantifier.

**QP** quantifier phrase.

**RNN** recurrent neural network.

**S** sentence.

**SEM** semantic interpretation of a syntactic head or phrase.

**SGD** stochastic gradient descent.

**SNLI** Stanford Natural Language Inference corpus.

**SNMT** Stanford Neural Machine Translation corpora.

**Spec,XP** specifier of the syntactic phrase XP in X’-theory.

**T** tense head (or any syntactic head in the T-domain).

**TP** tense phrase.

**UTAH** Uniformity of Theta Assignment Hypothesis.

**V** verb.

**VP** verb phrase.

**X’** intermediate (“bar-level”) phrase in X’-theory.

**XP** syntactic phrase headed by X.
Symbols

\( FN \) number of false negatives.
\( FP \) number of false positives.
\( P \) monadic predicate of type \(<e, t>\) (also: \( Q, W \)).
\( R \) dyadic relation of type \(<e, <e, t>\>\).
\( ReLu \) rectified linear unit.
\( S \) sigmoid function.
\( TN \) number of true negatives.
\( TP \) number of true positives.
\( X \) plural variable.
\( \downarrow \) operator over a predicate \( P \) such that \( \downarrow P \) applies to everything if \( P \) applies to nothing, and otherwise \( \downarrow P \) applies to nothing.
\( \Leftrightarrow \) logical equivalence.
\( \uparrow \) operator over a predicate \( P \) such that \( \uparrow P \) applies to everything if \( P \) applies to something, and otherwise \( \uparrow P \) applies to nothing.
\( \cos(\theta) \) cosine similarity.
\( \exists \) existential quantifier.
\( \forall \) universal quantifier.
\( \wedge \) logical conjunction.
\( \lor \) (inclusive) logical disjunction.
\( \preceq \) membership in the set of values of a plural variable.
\( \rightarrow \) logical implication.
$\sigma$ softmax function.

c logical constant.

e (i) Euler’s number ($\approx 2.71828$); (ii) singular first-order event variable.

tanh hyperbolic tangent function.

w possible world variable.

x singular first-order logical variable (also: y).

$<< e, t >>, << e, t >>, < e, t >>$ function from two monadic predicates to a monadic predicate.

$<< e, t >>, < e, t >>$ predicate modifier.

$< e, < e, t >>$ dyadic relation.

$< e, t >$ monadic predicate.

$< e >$ the semantic type entity.

$< t, < t, t >>$ function from two truth-values to a truth-value.

$< t, t >$ function from a truth-value to another truth-value.

$< t >$ the semantic type truth-value.
1. Introduction

Natural language processing (NLP) has undergone significant developments in recent years, mostly due to advances in machine learning (ML) techniques. Most state-of-the-art results in text classification, text generation, and machine translation are achieved using deep neural networks (DNNs) like recurrent neural networks (RNNs) [102, 183, 298], convolutional neural networks (CNNs) [308], and – more recently – Transformers [289]. These methods are grounded in algorithms that learn complex correlations between input features and the target classes (in text classification) or target sequences (in text generation).

In this dissertation I investigate the performance of multiple NLP techniques on classification, transformation, and representation of text data. The main focus is on use cases that have relevance for information security and privacy, but some results are more generally applicable to other NLP tasks. I refer to security- or privacy-sensitive use cases as adversarial, adopting the term neutrally with respect to whether the attacker’s or defender’s perspective is adopted. All five publications in the dissertation concern the English language.

From an adversarial perspective, the increased availability of automatic text analysis and generation methods constitutes both an asset and a threat. NLP can e.g. help detect criminal forum users [7], hate speech [14, 46, 76, 267, 292, 309], or troll accounts [50, 96, 198]. However, the possibility for text-based profiling or deanonymization by writing style constitutes a privacy breach, with potential consequences ranging from harassment to state-level punishment [11, 39, 40, 136]. Using NLP to identify the type of text content – content classification – also manifests potential for censorship and suppression of opinion.

To evade classification by content or writing style, the author can use text transformation to alter certain properties of the original text while

---

1 The term is used differently in the context of generative adversarial networks (GANs), where it means a generative model being trained to evade a classifier (Chapter 3; Section 3.1.2). Not all uses of GANs are adversarial in the security-relevant sense.
retaining others, with the goal of changing the classifier’s prediction but maintaining other aspects of interpretation for a human reader [95, 141, 176, 185, 240, 254, 271, 272, 301, 304, 310]. Such model evasion can be used both by an adversary to attack a legitimate classifier, as well as by a defender to prevent illegitimate profiling. Its success depends on three main aspects: (i) changing the relevant feature(s) of the original text; (ii) retaining other features of the original text that are not relevant for the classification; and (iii) the susceptibility of the classifier to be fooled by the transformation. While there are many ways to enact (i), maintaining (ii) at the same time is more challenging. With respect to (iii), a major question concerns how vulnerable different ML systems are to evasion; especially with respect to differences between model architectures (e.g. simpler ML techniques or DNNs) and input tokens (e.g. words or characters).

A matter that arises in all of the above – both classification and transformation tasks – is input data representation. In ML-based NLP, text is typically presented in a manner that is fairly close to a surface-level representation of the input string, such as a sequence of character or word n-grams. Simple ML techniques (Section 2.1.2) operate directly on such feature representations, whereas the hidden states of DNNs can combine input features in more complex and context-dependent ways (Section 2.1.3). However, detailed information retrieval tasks can require elaborate semantic representations of the input based on high-level theoretical abstractions. Prevalent formats include e.g. Discourse Representation Theory (DRT) [36, 139, 140], Abstract Meaning Representation (AMR) [19] and Minimal Recursion Semantics (MRS) [72]. Unlike data-based NLP frameworks, such approaches implement theoretical analyses concerning semantic structure, and use an explicit symbolic format for representation.

Theory-driven semantic representations and data-based NLP have largely developed separately from each other, although some methods of unification have been suggested [22, 86, 203]. Roughly, the former focus on constructing explicit representations of semantic structure for complex expressions; while the latter has been dominated by the distributional framework that represents units based on their relations to other units in the training corpus, focusing on the (sub)word-level. Vector representations of tokens produced based on their distributional properties are called embeddings. Combining these lines of research is useful; but a major dilemma is that theoretical analyses of semantic content tend to be complex, often involving dozens of semantic roles and relations that require significant expressive power from the metatheoretical formalism. Implementing such representations in a vectorized format (required for ML input) is highly non-trivial, and theory-driven work has mostly remained on the level of rule-based (symbolic) NLP. There is a thus a need to consider novel alternatives to make the approaches more compatible in practical NLP applications involving e.g. text classification and generation.
1.1 Research questions

This dissertation can be viewed through the dimensions of technical content on the one hand, and thematic content on the other. Technical content concerns NLP and ML methodology, covering the following classes with associated questions:

- **Text classification**: how well can NLP techniques detect the correct class of a text based on its content and/or writing style?

- **Text transformation**: how can certain targeted properties of a text be altered while maintaining its other properties to a maximal degree?

- **Text representation**: what are optimal formats for storing the meaning and grammar of a text for different NLP purposes?

Thematic content concerns the specific tasks the techniques are applied to, and covers the following classes with associated questions:

- **Adversarial text**: can NLP be effectively applied to security-relevant categories of text, such as deception or hate speech?

- **Author profiling**: are NLP techniques capable of detecting or profiling the author of a text based on the text alone?

- **Semantic representation**: how to represent semantic content for NLP purposes with minimal structure while retaining expressiveness?

The main classes of adversarial text discussed in this dissertation are deception and hate speech. Publication I surveys prior research on deception detection from text, as well as author profiling via writing style. Publication II and Publication IV provide original empirical results on the classification of hate speech. Publication II also demonstrates model evasion for hate speech classification via text transformation. This line of research is continued in a more elaborate manner for evading author profiling in Publication III. The matter of text representation arises throughout Publication I–Publication IV, for both classification and transformation tasks. Publication V focuses on this, presenting a novel format for semantic representation and applying it across multiple NLP tasks involving semantic parsing, information retrieval, text generation, and text transformation. Table 1.1 shows the allocation of each publication to the technical and thematic classes (allowing overlap).


**Table 1.1. Classification of publications in this dissertation.**

<table>
<thead>
<tr>
<th>Technical content</th>
<th>Thematic content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adversarial text</td>
</tr>
<tr>
<td>Text classification</td>
<td>Publication I</td>
</tr>
<tr>
<td></td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
<tr>
<td>Text transformation</td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
<tr>
<td>Text representation</td>
<td>Publication I</td>
</tr>
<tr>
<td></td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
</tbody>
</table>

1.2 Main results

For each cell in Table 1.1, I will defend the following high-level conclusions, divided among Chapters 2–4:

**Text classification (Chapter 2):**

- **Adversarial text:** classifier performance in detecting deception or hate speech is more dependent on training data than model architecture (Publication I; Publication II; Publication IV).

- **Author profiling:** *stylometry* enables the identification or profiling of authors with high performance, which constitutes a genuine privacy risk for the authors (Publication I; Publication III).

- **Semantic representation:** an appropriate representation format allows conducting effective and very detailed text comparison in terms of semantic and grammatical structure, which can be used for e.g. *parallel corpus construction* (Publication V).

**Text transformation (Chapter 3):**

- **Adversarial text:** hate speech classifiers are easily evaded by simple text transformation attacks that require neither white- nor black-box access to the targeted model (Publication II).

- **Author profiling:** *writing style transfer* is possible via either rule-based techniques or encoder-decoder networks; but only the former are able to retain semantics to a sufficient degree for practical application (Publication III).
– **Semantic representation**: encoder-decoder-networks are unreliable for maintaining semantic content in writing style transfer (Publication III); but separating grammatical features and semantic content in a single vectorizable input representation allows conducting *grammatical transformation* (Publication V) in an effective manner.

**Text representation (Chapters 2–4):**

– **Adversarial text**: features relevant for detecting deception (Publication I) or hate speech (Publication II; Publication IV) are *too simple* to correspond meaningfully to human evaluation.

– **Author profiling**: low-level features (e.g. character n-grams) fare the best overall in stylometry; but the relevant features may not be genuinely stylistic as opposed to being sensitive to semantic content (Publication I; Publication III).

– **Semantic representation**: effective representation of semantic content is possible with far fewer semantic roles and simpler structure than in standard formalisms applied in NLP (Publication V).

### 1.3 Outline

Chapters 2–4 are organized around the technical content classes in Table 1.1. Chapter 2 focuses on text classification, Chapter 3 on text transformation, and Chapter 4 on text representation. Of these, Chapters 2 and 3 are internally structured by the thematic classes in Table 1.1, and cover all five publications in the dissertation. In contrast, Chapter 4 centers around semantic representation and mostly discusses Publication V. While the other publications also have relevance for text representation, this discussion is allocated to the previous chapters since it is so tightly related to their dedicated technical classes (text classification or transformation). Chapter 5 summarizes the main results of the whole dissertation, and suggests possible directions for future work.
2. Text classification

In this chapter I review the ML techniques used for text classification across the publications contained in this dissertation (Section 2.1), and classification results on four types of categories: deception, hate speech, author class, and semantic/grammatical structure (Sections 2.2). All five publications are addressed, but discussion is centered around especially Publication I, Publication II, and Publication IV.

2.1 Features and classifier architectures

All classification is ultimately based on finding correlations between target classes and input features. Success is measured by metrics that compare classifier predictions with gold standard labels, provided by human labelers or some other source considered sufficiently reliable. A simple metric is accuracy, which gives the ratio between correct predictions and all predictions (Definition 1).

Definition 1 (Accuracy).

\[
\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP}
\]

where \(TN\) is the number of true negatives, \(TP\) of true positives, \(FN\) of false negatives, and \(FP\) of false positives.

Accuracy can be unreliable especially with imbalanced datasets. For example, if 90% of the data belongs to class 1 and 10% to class 0, achieving 90% accuracy would be trivial simply by always predicting class 1. More appropriate measures are given by precision and recall (for each class), the harmonic mean of which is the F1-score (Definition 2).

Definition 2 (Precision, recall, and F1-score).

\[
\text{Precision} = \frac{TP}{TP + FN}
\]
Next, I review the input features (Section 2.1.1) and classifier architectures (Sections 2.1.2–2.1.3) that have a prevalent role in this dissertation.

2.1.1 Features

One of the most important aspects of designing a classifier is selecting the input feature type: which features are included, how much pre-processing is involved, etc. In addition to classification performance, the choice of features can have large impact on e.g. model susceptibility to adversarial examples. DNNs typically use simple features like characters or (sub)words, while more complex hand-crafted features tend to be relevant for simpler ML techniques.

Characters
The most basic feature choice in text classification is to use each character as a separate feature. A major benefit of character features is that they do not require any pre-processing and are consequently independent of the input language. Their main drawback is that they lack semantic content, which makes it difficult for humans to interpret their impact; even when using classifier architectures that allow the human-readability of feature relevance (such as logistic regression; see Section 2.1.2).

Simple classifier architectures (Section 2.1.2) lack the capacity for representing complex context-dependencies between tokens, which can limit their performance when using only single characters as features. N-grams are tuples of multiple tokens that appear in succession, and can be used to capture more intricate relations between characters. Unigrams are single tokens, bigrams have two tokens, trigrams three, etc. Skip-grams are n-grams that allow other elements to intervene between the tokens.

Words
As opposed to characters, words have (semi-)independent semantic content,\(^1\) which makes them optimal for human-interpretation. In languages like English, word tokenization is relatively simple even by whitespace

\(^1\)Of course this statement is not literally true, when phenomena like homonymy, polysemy and contextual influences on interpretation are considered. Nevertheless, the compositionality of linguistic meaning requires at least some level of independence between the semantic contents of words, in order for complex expressions to be assigned meaning as a function of their parts (see Chapter 4).
alone because word boundaries are clearly marked. Nevertheless, typically a more complex tokenization algorithm is used, such as those provided by NLTK [29] or Spacy [127]. They separate clitics like the possessive ’s as separate words, decreasing the vocabulary and hence increasing model generalizability. Like with characters, n-grams can be used to capture contextual dependencies between adjacent words. For languages with complex morphology where words are less clearly grammatically marked, word tokenization is more challenging [152].

More complex pre-processing stages for words are their lemmatization or stemming. The purpose of both is to assimilate different inflectional variants of the same word. Their benefit depends on the task at hand, with more abstract semantic representations likely benefiting from such assimilation (see Chapter 4) while other tasks make use of inflection information (e.g. stylometry; see below).

**Subword tokens**

As a middle-ground between words and characters, subword tokens are derived based on the most common character n-grams that appear in a text corpus [158, 270]. They increase the applicability of features to novel words while retaining part of the semantic interpretability of words. State-of-the-art pre-trained DNN models like BERT [81] and its derivatives [133, 169, 175, 264] use subword tokens as input features.

**Embeddings**

Instead of operating directly on input tokens, DNNs typically have an embedding layer, which maps each input token to a fixed-length real-valued embedding vector. An embedding matrix contains embedding vectors for each token in the vocabulary as rows in the matrix. The most prevalent embedding algorithms are word2vec [200] and GloVe [225] (see Chapter 4 for more discussion). Pre-trained embedding matrices also exist, and can be used for initializing embedding layer weights, or for tasks like word similarity evaluation via vector distance metrics. In addition to words, subword tokens can also be represented as embeddings. Pre-trained subword embeddings are provided e.g. in the BPEmb library [123].

**Grammatical features**

Tasks like stylometry – text classification based on writing style – often use features that are less relevant for semantic content and more for grammatical structure. Such features come in many kinds, and usually require pre-processing such as part-of-speech (POS) tagging or syntactic parsing. Possible features include e.g. POS n-grams, dependency tags (see Chapter 4), and punctuation. The Writeprints feature set [312] incorporates multiple grammatical features, and has been applied in many stylometry studies [1, 6, 11, 88, 193, 217]. Table 2.1 shows examples.
Table 2.1. Examples of the Writeprints features (270 altogether) [312].
(Table modified from Publication I.)

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Example features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>Character: number of characters, number of letters, number of digits, frequency of letters, frequency of special letters</td>
</tr>
<tr>
<td></td>
<td>Word: number of words, average word length, vocabulary richness, average sentence length</td>
</tr>
<tr>
<td>Syntactic</td>
<td>frequency of punctuations, frequency of function words</td>
</tr>
<tr>
<td>Structural</td>
<td>number of sentences, number of paragraphs, number of sentences/words/characters in a paragraph, presence of quotes</td>
</tr>
<tr>
<td>Content-specific</td>
<td>frequency of content specific keywords</td>
</tr>
</tbody>
</table>

2.1.2 Simple machine learning architectures

Moving on from features to classifiers, this section presents the simple (non-DNN) ML architectures that play a significant role in this dissertation. (The range of all possible techniques is thus greater than reviewed here.)

**Logistic regression**

Logistic regression (LR) minimizes the *logistic loss* (LogLoss; also known as *cross-entropy loss*: Definition 3) between true and predicted labels.

**Definition 3** (Logistic loss).

\[
\text{LogLoss} = -\frac{1}{n} \sum_{i=0}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]
\]

where \( y_i \in \{0, 1\} \) is the correct label, and \( \hat{y}_i \in [0, 1] \) is the prediction.

LR can be thought of as a simple neural network with no hidden layers, where each input node’s activation is multiplied by its *weight*, and all these are summed and put through the logistic *sigmoid function* \( S \) that yields a value in the range \([0, 1]\) (Definition 4). The network is then trained to minimize the LogLoss in the training set by updating the weights.

**Definition 4** (Sigmoid function).

\[
S(x) = \frac{1}{1 + e^{-x}}
\]

where \( e \) is Euler’s number \( \approx 2.71828 \).
Decision trees and random forest

Decision trees are non-linear predictive models that divide the data into subsets at multiple steps based on the input features. At each step, the subset division is based on the input feature that maximizes the so-called purity of the nodes: the less class overlap there is between the nodes, the purer they are. A common metric to use is the Gini index, which is a number in the range $[0,1]$ from purest to least pure (Definition 5). As an impurity measure, Gini provides the smallest value for the purest node to be chosen for the next subset division in the tree.

**Definition 5** (Gini index).

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2$$

where $P_i$ is the probability of the datapoint being assigned to class $i$.

Decision trees have a tendency to overfit the training set, which can deteriorate performance on external test sets. One way to alleviate this problem is random forest, which is an ensemble ML technique that makes its decision based on a majority voting scheme (e.g. averaging) across multiple decision trees trained on different subsets of the training set.

Multilayer perceptron

Contemporary deep learning builds on prior neural network models, of which the perceptron algorithm was the earliest incarnation [259]. Unlike nodes in a DNN, the perceptron has a discrete threshold activation function, which yields either 0 or 1 based on whether or not the weighted sum of its inputs exceeds the threshold value (such as 0: Definition 6).

**Definition 6** (Perceptron activation function).

$$f(x) = \begin{cases} 
1, & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\
0, & \text{otherwise} 
\end{cases} \quad (2.1)$$

where $x$ is the input vector, $x_i$ is the input of the $i$:th input node, $w_i$ is the weight between the $i$:th input node and the perceptron, and $b$ is the bias.

The neural network consists of perceptrons trained to minimize the loss between true and predicted labels by adjusting the weights and bias. Famously, the addition of a hidden layer made it possible to implement exclusive disjunction (XOR), and – more generally – a truth-functionally complete set of logical connectives [261]. Therefore at least one hidden layer is used; hence the name multilayer perceptron (MLP).
2.1.3 Deep neural networks

In comparison to MLPs, DNNs use contiguous activation functions, which makes their loss functions differentiable with respect to modifiable parameters across the network (weights and bias terms).\(^2\) This, in turn, allows the use of gradient descent for minimizing the loss by modifying the parameters during training. Standard DNNs are fully connected, like MLPs. A basic fully connected DNN pipeline is shown in Figure 2.1. Here, every input node gives its weighted activation to every output node, which sums its inputs together and applies an activation function. Common activation functions are sigmoid (Definition 4 above), hyperbolic tangent \((\text{tanh}: \text{Definition 7})\) and rectified linear unit \((\text{ReLU}: \text{Definition 8})\).

**Definition 7** (Hyperbolic tangent = tanh).

\[
\text{tanh}(x) = \frac{e^{2x} - 1}{e^{2x} + 1}
\]

**Definition 8** (ReLU).

\[
\text{ReLU}(x) = \max(0, x)
\]

Depending on the task, target values can be binary (classification) or real-valued (regression). In classification, the softmax function \((\sigma: \text{Definition 9})\) is typically applied to the final layer, which yields a probability distribution across the output units that represent the target classes.

**Definition 9** (Softmax).

\[
\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}
\]

where \(x\) is the input vector, and \(K\) is the number of output classes.

\(^2\)The ReLU function (Definition 8) is not differentiable at the origin, where the derivative can be manually set to zero in practical implementation.
A loss function – such as LogLoss (Definition 3) – then calculates the loss based on this probability distribution and the target class value. The differentiability of the activation functions allows tracking the gradient (vector of partial derivatives) of the loss with respect to the model parameters. Based on this, the effect of each parameter on the loss is calculated, which is called the back-propagation of error. The parameters are then adjusted to minimize the loss, which is called gradient descent. The updates are further controlled by multiplying them with the learning rate, which is a hyperparameter (i.e. assigned to the model explicitly).

Batch gradient descent calculates the gradient for the whole dataset, after which the update is done. In stochastic gradient descent (SGD) multiple updates are done for randomly selected datapoints, and in minibatch gradient descent for multiple subsets of the training data. DNNs are usually trained using minibatches, but the term “SGD” is often used for this in the literature as well. Additional optimization procedures have been developed to make training faster, such as Momentum [244], AdaGrad [82] and AdaDelta [307]. In this dissertation, all DNNs that we designed and trained ourselves used the Adam optimizer [151].

**Convolutional neural network (CNN)**

CNNs [164] are specifically tailored for image classification, but can also be applied for text [308]. Their main purpose is to apply feature detectors across different positions in the input. The input is first mapped to a convolution layer, which contains kernels (or filters) that are weight matrices of a fixed dimensionality. Kernels are smaller than the input itself, and the same kernel is applied across the input via weight sharing between nodes that have different receptive fields. Weight sharing gives CNNs the important property of translation invariance: kernels are sensitive to the relevant input feature configuration anywhere in the input, which makes the network less susceptible to data perturbations.

CNNs typically pool the output of a group of convolutional units to a single unit, e.g. by choosing their highest value (max pooling) or average.
Text classification

![Figure 2.3. Vanilla RNN.](image)

\(x_i\): input at step \(i\)
\(h_i\): hidden state at step \(i\)
\(y_i\): output at step \(i\)
\(U\): weights between input layer and hidden layer
\(W\): weights between previous and current hidden layer
\(V\): weights between hidden layer and output layer

(average pooling). Complex CNNs can incorporate multiple convolutional and pooling layers by applying subsequent convolution on the outputs of earlier pooling layers, in effect implementing higher-level feature detectors. At the final classification stage, CNNs use a fully connected layer (and softmax) like standard DNNs. Figure 2.2 displays a basic CNN pipeline.

Recurrent neural network (RNN)

Sequence-to-sequence mapping can be thought of as repeated application of a DNN to each token in the input sequence. However, simple DNNs are unable to take contextual information into account, as they always map the same input token to the same output token regardless of its surrounding environment. RNNs introduce an additional input source: the previous hidden state of the network.

In basic (“vanilla”) RNNs, the input for the network at timestep \(i\) is the current input token combined with the hidden state from the previous timestep \(i - 1\), as shown in Figure 2.3. This way, the same input token can result in a different output depending on the prior hidden state, which in turn is influenced by prior inputs and hidden states. After each input token the RNN produces an output for the corresponding timestep, and the final output functions as the representation of the entire input sequence.

The most notable limitation of vanilla RNNs is the vanishing gradient problem, where the effect of earlier input tokens decreases in later stages of the sequence [125]. More sophisticated variants of RNNs have been developed to address this issue, most notably long short-term memory networks (LSTMs) [126] and gated recurrent units (GRUs) [64], shown in Figures 2.4a and 2.4b.

**Long short-term memory network (LSTM).** As an addition to the standard RNN architecture, LSTMs include another hidden state, called the cell state. In contrast to the standard hidden state, the cell state is updated
Figure 2.4. LSTM and GRU cells.

(a) LSTM  
\[ \begin{align*} 
    x_i & : \text{input at step } i \\
    h_i & : \text{hidden state at step } i \\
    c_i & : \text{LSTM cell state at step } i \\
    \times & : \text{element-wise multiplication} \\
    1 - & : \text{element-wise subtraction from 1} \\
    \sigma & : \text{sigmoid function (Definition 4)} \\
    \tanh & : \text{hyperbolic tangent function (Definition 7)} 
\end{align*} \]

(b) GRU

in a way that better allows the maintenance of information across long sequences. At each iteration, a *forget gate* is first applied to the cell state to determine how much of the information is erased from the prior state. Additional gates are then applied to the current input to determine its effect on the cell state. Long-distance dependencies thus become better detectable by the forget gate learning to retain information in the cell state (i.e. not erase it). The cell state also influences the standard RNN hidden state and vice versa.

**Gated recurrent unit (GRU).** LSTMs increase model complexity by doubling hidden states. GRUs avoid this by only using one hidden state, which combines the functions of the standard RNN hidden state and the LSTM cell state. So-called *update gates* and *reset gates* condition the influence of new input to the prior hidden state, allowing the retainment of old information across multiple iterations. Comparative evaluation across multiple sequence modelling tasks has demonstrated GRUs to be on par with LSTMs, although not quite as effective [64].

**Attention.** Without attention, RNNs make the classification decision only based on the final hidden state. Attention allows this decision to depend on prior hidden states as well. A list of prior hidden states is kept in memory, and the attention mechanism learns to assign probability weights to them based on the current input (including the influence of prior hidden states). A weighted combination (e.g. sum or multiplication) of prior hidden states then functions as additional input to the final layer. Attention thus increases the influence of the most relevant tokens in the input sequence. Its addition has markedly improved the success of RNNs [183].\(^3\)

\(^3\)The attention mechanism can be thought of as a DNN-based implementation of a query-key-matching operation to produce the value corresponding to the key. In a standard query system, a query is assigned to the key it most resembles,
Transformers

Discarding recurrence, Transformer networks are solely built around attention [289]. They have a multi-headed self-attention layer, which determines how much attention is given to each input word for their respective contextual encodings. This allows surrounding words to effect the encoding in addition to the word itself. Self-attention is followed by a fully connected layer. Many iterations of the self-attention and fully connected network can be applied in sequence, where each new self-attention module takes the outputs of the previous fully connected layer as inputs. In this respect Transformers partly resemble CNNs with hierarchical convolutions, but with Transformers using self-attention instead of convolution.

Transformers are designed as encoder-decoder architectures for sequence-to-sequence modelling, which is why I explain their structure in more detail in Chapter 3. However, large pre-trained Transformer networks like BERT [81] can also be used for text classification by fine-tuning them to a classification task on a separate training set.

2.2 Thematic content

Following Table 1.1, I divide the thematic classes into adversarial text, author profiling, and semantic representation. In this section I go through the main questions concerning these topics in relation to text classification, and address them based on results from Publication I–Publication V. I formulate three major questions as follows, related to each thematic class:

- **Adversarial text**: how effective are state-of-the-art classification systems at detecting (i) deception, and (ii) hate speech; and what explains their success (or the lack of it)?

- **Author profiling**: how effective are state-of-the-art stylometry applications at deanonymizing authors, or profiling them based on privacy-sensitive information (e.g. gender or age)?

- **Semantic representation**: how can text be classified based on detailed information about semantic and/or grammatical features, and what repercussions can this have?

The key is then assigned to its value in a database. In neural attention, the queries, keys, and values are all matrices, and the query-key matching process is some similarity metric between the query vector and each key in the key matrix. Traditional key-value pairing would thus be achieved by using a one-hot similarity metric between the query and the keys. Applying softmax over this similarity vector results in a probability distribution across the input positions, and the value for the query then corresponds to the weighted sum of all possible values with these weights.
In each case, I place particular emphasis on security and privacy considerations. In addition to reviewing the main results from Publication I–Publication V, I give my evaluations on the most urgent issues to focus on in future work on related topics. Section 2.2.1 focuses on adversarial text, Section 2.2.2 on author profiling, and Section 2.2.3 on semantic representation.

2.2.1 Adversarial text: deception and hate speech

This dissertation concentrates on two main types of adversarial text, which I center this section around: deception (Publication I) and hate speech (Publication II; Publication IV). Other classes – such as spam or trolling – are briefly discussed in Publication I, but to a lesser degree.

Deception

Many types of adversarial text belong under the umbrella term of deception. It is a broad concept, which makes its scientific operationalization challenging. Nevertheless, a common thread can be detected in the literature: deception is an aspect of speaker intention, where the speaker says something with the deliberate attempt of making the hearer believe something the speaker believes to be false [9, 44, 80].

Crucially, the deceptiveness of text differs from its factuality. There are at least two ways in which a factually correct expression can be deceptive. First, the speaker can falsely believe it to be factually incorrect, and try to make the hearer believe it despite this. Second, the speaker can use (what they believe to be) a true expression in an attempt to indirectly convey a false implicit message. The latter possibility relies on the inferential character of communication, where instead of only semantically decoding the expression, the hearer draws various conclusions from it [52, 110, 277]. Such implicit inferred content often constitutes the most important aspect of the message the speaker actually wants to convey. The (believed) factuality of such intended messages – not only the expression itself – should be the foundation of a theoretically satisfactory account of deception.

The character of deception as a type of speaker intention constitutes at least a prima facie problem for its detection via text classification. While we might expect deceptive content in certain contexts to correlate with various linguistic properties, a more controversial question is whether such correlation applies across different contexts. The main theoretical motivation for this idea is based around the same notion that motivates physiological polygraph tests: detecting indirect consequences of emotional responses to deception, such as increased stress and anxiety. Accordingly,

---

4I use the terms “speaker” and “hearer” in a modality-independent way, encompassing both written and spoken text. All empirical results I discuss here and Publication I are derived from written text.
Table 2.2. The most common linguistic cues to deception (modified from Publication I).

<table>
<thead>
<tr>
<th>Studies</th>
<th>Cue</th>
</tr>
</thead>
<tbody>
<tr>
<td>[45, 119, 144, 171, 211, 215, 216, 313, 314]</td>
<td>High emotional load</td>
</tr>
<tr>
<td>[165, 171, 179, 215, 306]</td>
<td>High use of first-person pronouns</td>
</tr>
<tr>
<td>[118, 144, 197, 211]</td>
<td>Low use of first-person pronouns</td>
</tr>
<tr>
<td>[73, 215, 216, 315]</td>
<td>High use of verbs</td>
</tr>
<tr>
<td>[119, 165, 197]</td>
<td>High use of certainty-related words</td>
</tr>
</tbody>
</table>

deceivers might exhibit e.g. more negatively laden terminology without realizing this [84]. If this was empirically corroborated, it would give support for the possibility of deception detection independent of context, at least to some degree.

In the first half of Publication I we argue that the hypothesis of context-independent deception markers is not empirically supported. Reviewing prior work on NLP-based deception detection, we conclude that no linguistic features correlating with deceptiveness as such are found. Instead, the features that have been suggested to play such a role can be more readily accounted for by their connection to discourse-specific semantic content that deceivers tend to emphasize. Importantly, such features should not be assumed to generalize across datasets. This points to a notable deficiency in only relying on a single dataset per study.

Table 2.2 collects the deception-indicative features reviewed in Publication I that appear in at least three independent studies on different datasets. Three major observations can be made. First, some features are manifestly contradictory: both the high and low use of first-person pronouns (I, we) correlated with deception, but in different datasets. Also, while high emotional load was the most recurring feature type, sentiment (positive/negative) varied based on the dataset. An excess of negative terminology correlated with deceptive texts that had negative sentiment, but not texts with positive sentiment where the effect was the opposite [215, 216]. This indicates that the correlation was due to the semantic content of the deceptive text, not its deceptive nature as such. Hence, even if the hypothetical stress effects existed, they have not had reliable observable effects on experimental outcomes.

Second, the features are clearly indicative of semantic content in ways that might correlate with deception, but are not as such constitutive of it. For example, the use of general and abstract words could be accounted for by the general lack of information available to the deceiver; but the observation was no longer made in contexts where such information was better available. As Ott et al. [215] point out, in fake reviews the author can often have access to much information about the object of the review in spite of
lacking actual first-hand experience. As expected, they found that in fake reviews, generality was not indicative of deception; indeed the opposite was the case. This illustrates that generality is only indirectly related to deceptiveness, and can be divorced from it depending on the context. A similar explanation is also available for the seeming contradiction between both high and low uses of first-person pronouns correlating with deception. In different contexts, deceivers can either avoid a personal perspective or make use of it, depending on its (assumed) effects on the hearer.

Third, the features are far too generic to be reliable for deception detection: any combination of them can appear in non-deceptive text as well. A proper deception detection system should be able to distinguish between deceptive and non-deceptive variants of otherwise similar texts. Instead, the features in Table 2.2 are common lexical correlates of deceptive as opposed to non-deceptive text. They indicate what kinds of contents deceivers tend to talk about, not how deceptive and non-deceptive texts with similar contents would differ. Hence, even if their correlation with deception was clearly above random in some test setting, this would not warrant their use for suspecting deception in another setting.

In terms of classifiers, most studies reviewed in Publication I use simple ML techniques rather than DNNs. The main benefit of simple ML techniques is their human-interpreatability: in e.g. LR the weight of each feature on the classification is accessible for evaluation. This way, results like those summarized in Table 2.2 can be uncovered; whereas with DNNs there are no direct means of obtaining such information. While some comparative evaluation has been made between different classifiers used for deception detection [314], Publication I does not reveal any overarching correlations between classifier architecture and performance. Given the fundamental problems discussed above, it is unlikely that more complex DNNs would significantly improve the situation.

The main message we draw in Publication I is that deception as such is not manifested in any linguistic features, but can only correlate with deception in highly context-dependent ways. While this is in principle a negative result, it does not invalidate any of the studies reviewed there. Rather, I would argue that such research is more appropriately interpretable as corpus analysis of deceptive texts across different domains. As such, it provides important empirical information about the manifestation of deception in a variety of text types and use cases, and can play a vital role in furthering our understanding of deception as a communicative act. It does not, however, provide a reliable method for detecting deception using only linguistic features.

One alternative method for indirectly detecting deception-related material via NLP is to use similarity metrics between texts. This does not detect deceptiveness as such, but can be of help in related tasks like troll or spam detection [163, 198, 209], fake news detection [214, 227] or rumor
debubking [260]. Such techniques can further combined with metadata about e.g. user behavior in discussion forums [199], which I refrain from discussing here because it goes beyond NLP. For obtaining detailed information about text content, it can be beneficial to use more elaborate means of *semantic representation* than basic word- or character features, as discussed in Section 2.2.3, Chapter 4, and Publication V.

**Hate speech**

Moving on to the second major adversarial text type in this dissertation, Publication II and Publication IV concern *hate speech*. While the line between hate speech and merely “offensive” content is difficult to draw [76], overt expressions of hate are often linguistically tractable. Direct threats of violence, explicit dehumanization of (groups of) people etc. can uncontroversially be considered as hate speech, if the term is to be used at all. At least in such clear cases, hate speech does not suffer from as severe difficulties of operationalization as deception. However, it is important to keep in mind that drawing the boundary between acceptable and unacceptable speech is never trivial, and always has political implications that extend beyond technical considerations [42, 51].

Proposed systems for hate speech detection are manyfold, both from the academia and industry [14, 38, 46, 76, 243, 267, 287, 300, 309]. Like in deception detection, a major problem in this line of research has been the lack of proper *comparative evaluation* across datasets and model architectures. Additionally, as a security application, hate speech detection should be subjected to *adversarial examples* that deliberately aim at evading classification. Publication II addresses these matters by examining proposed systems in terms of *generalization* and *robustness*.

We conducted a comparative evaluation across multiple hate speech classifiers proposed in prior research. In model architecture, the classifiers ranged from simple LR or MLP [76, 300] to complex DNNs: an LSTM [14] and a combination of a GRU with a CNN [309]. Some (simple) models used characters as input features, and the rest (including all DNNs) used words. The original models were derived from these architectures with four datasets, three derived from Twitter and one from Wikipedia edit comments. One Twitter dataset had three classes, distinguishing between hate speech, offensive but non-hateful speech, and ordinary speech [76]. We made another two-class dataset from this by combining the ordinary and offensive classes into one, yielding four two-class datasets and one three-class dataset altogether.

The first main observation was that when trained on the same datasets, all two-class models performed similarly regardless of architecture. In

---

5In this section, I make no distinction between hate speech and *toxic speech* because the datasets used in Publication II and Publication IV all fall in both categories, but such differences can be relevant in other contexts.
particular, there was no correlation between model complexity and performance. This was most likely because the relevant features in all datasets were simple enough to be detectable even by basic ML architectures, and DNNs had no reason to exploit their more complex capacities beyond simple feature detection. Likely, all classifiers got caught on particular words correlating with the “hate” class, such as curse words. This hypothesis received further support from the observation that when tested on “offensive” class texts from the three-class dataset, the two-class models usually assigned them to the “hate” class.

Similar model performance despite vast architectural differences demonstrates an important issue: while state-of-the-art DNNs have the capacity for detecting complex feature relations, nothing guarantees that they actually use these capacities if simpler features are dominant. Especially in small datasets, basing classification on simple features (like the presence of curse words) was readily available, and increasing classifier complexity did not seem to influence this to one direction or another.

The second major finding in Publication II was that when classifiers were trained on (the training set of) one dataset and tested on (the test set of) another dataset, all classifiers failed to reach decent performance. This indicates that all four datasets differed in which features were the most relevant for classification, despite all exemplifying some type of hate speech. This problem further illustrates the importance of training data as opposed to model architecture as the main factor for model performance.

Finally, we showed that the classifiers were susceptible to evasion via simple attacks, assuming neither white- nor black-box access for the adversary. These attacks used word-internal changes (e.g. typos), word boundary changes (adding or removing whitespace), and the addition of unrelated non-hateful words. I discuss them more thoroughly in Chapter 3, since they make use of text transformation. A largely beneficial mitigation technique was adversarial training, where the same kinds of attacks were added to the training set.

In summary, a major finding of Publication II was that hate speech classification performance depended mostly on the training dataset rather than model architecture. It is therefore unlikely to achieve significantly better results merely by complicating the ML algorithms alone. Instead, it remains necessary to increase training data in both size and variability. This presents a challenge especially for more specific subtypes of hate speech (e.g. threats), which are only available in small quantities [287].

In Publication IV we used data augmentation for improving hate speech classification with a small and highly imbalanced seed dataset. We used Kaggle’s Toxic Comment Classification Challenge dataset, and concentrated on binary classification with the “threat” class as the hate speech

---

\[6\text{https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge}\]
variant. Since the augmentation techniques involve text transformation, I review them in more detail in Chapter 3. Here, I instead focus on the classifier architectures we compared in these experiments.

As simple ML models we applied two LR classifiers: one using character n-grams and the other word n-grams as input features. As DNN models, we trained a word-based CNN from scratch; and fine-tuned BERT [81], which is a large pre-trained Transformer network. Like in Publication II, we were particularly interested in comparing the classification performance between simple architectures (LR here) and complex DNNs.

An important difference between fine-tuned pre-trained networks (like BERT) and models trained on the small seed data alone is that pre-training gives the model prior information to utilize in learning the relevant features for the task it is fine-tuned on. This is called transfer learning, since information contained in the pre-trained network is transferred to the new task. The problem of feature simplicity in training data (discussed above) is less likely to be detrimental for transfer learning, because the model has already stored much linguistic information that it might make use of even if it is not directly present in the seed training set.

As expected, we achieved the best classification results with BERT; although this required augmenting the data at least by oversampling the minority (“threat”) class (see Chapter 3; Section 3.2.1). In contrast, the CNN did not outperform the simple LR classifiers, despite having more complex structure. Likely, the success of BERT was due to its pre-training on a very large dataset that incorporates general information about the co-occurrence of (sub)words in English [81]. Furthermore, we were able to closely approximate BERT’s performance with the simple LR models when using more elaborate data augmentation. On the character-LR classifier, we remained within only 1% difference from the best F1-score attained with BERT (0.70 vs. 0.71), which was not a statistically significant difference across 20 repetitions with different randomly sampled seed datasets (using one-sided paired t-tests with $\alpha = 5\%$). This finding further supports the hypothesis that the main factor in determining classifier success was training data rather than model architecture.

A common conclusion can be drawn from both Publication II and Publication IV: collecting more training data is the main requirement for improving hate speech classification, as merely complicating the classifier architecture has shown little to no positive effects on the datasets studied. Publication IV demonstrates the potential of automatic augmentation techniques for obtaining additional training data, at least for improving extremely low initial classification performance. Special attention should thus be given to increasing the variability of the initial seed training data, since automatic data augmentation can be used to create many similar variants of original datapoints without manual labeling.

For improving our understanding on why different datasets seem to
The third main text classification task discussed in this dissertation is author profiling based on textual features alone. Unlike the detection of deception or hate speech, such profiling should be sufficiently independent of semantic content, indicating only the writing style of the author. However, in practice, this stylometric profiling uses both grammatical and lexical features. If target classes are conceptualized as sets of authors, deanonymization is a special type of author profiling where the target classes are singleton sets each containing one author. In other cases, the class can contain multiple authors, based on e.g. gender or age [254, 266].

In the second half of Publication I we review the literature on author deanonymization via stylometry, focusing especially on its status as a privacy risk to deliberately anonymous authors. Brennan and Greenstadt [39] coined the term adversarial stylometry to designate scenarios where stylometry is used by an adversary to deanonymize an author against their will. They discuss an imagined case of an employee who files an anonymous complaint against her employer, who then uses stylometry to identify her. State-level penalties for whistleblowers are also major concerns in countries that have strict censorship laws.

The main question in evaluating the severity of the privacy risk posed by stylometry is whether authors can reliably be detected by writing style alone, or whether classification success is highly limited (as in deception detection; see Section 2.2.1). In Publication I we argue that the risk posed by adversarial stylometry is real, and the success of author deanonymization is far greater than that of e.g. deception detection. However, the distinction between writing style and semantic content has often not been drawn clearly, and requires further scrutiny in future work.

In their survey, Neal et al. [210] make the judgement that author deanonymization can generally succeed well when the number of authors is relatively low. At least with 20 authors a very high accuracy has been received (97.69%) [312]. However, significant results have been obtained even with far higher author counts: e.g. Narayanan et al. [208] achieved over 20% accuracy on 100000 authors. While this is insufficient for satisfactory deanonymization, it is a marked increase over the initial probability with
random selection (0.001%). From the perspective of the defender, such a large increase in the likelihood of deanonymization already poses a notable privacy risk. Furthermore, an initial stylometric classifier could first be applied to reduce the number of the most likely authors to a more manageable size for further manual and/or automatic evaluation (e.g. ≤ 20).

While DNN-based stylometry has become more prominent in recent years [15, 41, 100, 279], most studies have used traditional ML techniques with feature engineering extending beyond the use of character or word n-grams. The Writeprints feature set (Table 2.1) is a prominently applied collection of character-based, word-based, and grammatical features used for author deanonymization [1, 6, 11, 88, 193, 217, 312]. However, its success alone does not reveal which features are the most relevant.

In Publication I we review prior research comparing the success of different individual features in stylometry, reaching the overall conclusion that character n-grams tend to have uniformly strong performance. In contrast, higher-level features like vocabulary richness or average sentence length have had less success [111]. This is somewhat problematic, given that the latter can be expected to be less dependent on semantic content, and hence more genuinely stylistic. In contrast, character n-grams strongly correlate with lexical items and therefore semantic content, resulting in the classification becoming more based on content rather than style in a strict sense. The increased prevalence of DNNs makes this issue even more pertinent, as DNNs typically use characters or (sub)words as input tokens, which makes the classification more likely to be based on lexical content. As more genuinely stylistic features, function words and punctuation have shown success across many studies [63, 135, 206, 284, 311].

The problem of separating style from content is also evident in author profiling beyond deanonymization, across classes that contain many authors. For instance, some studies have treated political stance (e.g. Republican or Democrat) as classes detectable via “writing style”, even though the relevant features are largely content-based [240]. For this reason, author profiling might be better described as text classification more generally, rather than stylistometry as such. An author might be recognized not due to their writing style but due to their propensity to discuss certain topics. This is especially relevant for mitigating author profiling via style transfer (Chapter 3; Section 3.2.2). Style transfer can effectively evade the classifier only to the extent that relevant features are genuinely stylistic.

DNNs have become more common in more recent stylometry research [15, 41, 100, 279], but traditional ML models remain the most common in prior work overall [1, 6, 11, 88, 111, 135, 193, 217, 239, 210, 312]. Like with deception detection (Section 2.2.1), no overarching consensus exists on the effectiveness of particular classifier architectures above others. For example, support vector machines have worked well with the Writeprints features on the Extended Brennan-Greenstadt corpus [39], but Mahmood
et al.’s [185] comparison on the same data favoured random forests. In our evaluation of multiple ML architectures across four datasets in Publication III, MLPs performed the best with the Writeprints features. Model architecture choice remains an important part of experiment design; but prior results on different datasets are insufficient to warrant any general conclusions on which architecture is optimal for any particular test setting.

In Publication III we further conducted novel experiments on multiple datasets. Across four two-class datasets on author profiling (gender or age) and deanonymization (between two blog authors or between Donald Trump and Barack Obama), we used three classifiers: a word-based LSTM, a word-based CNN, and a Writeprints-based MLP. Both DNNs outperformed the Writeprints classifier in most experiments. Likely, the most relevant features had to do with the prevalence of particular words in certain combinations, which the DNNs were tailored for. This illustrates the challenge discussed above: much of what constitutes “style” especially in DNN-based stylometry seems to concern lexical choices rather than strictly stylistic aspects of text.

I draw two main overall conclusions on author profiling. First, profiling/deanonymization is possible with low-level linguistic features like character or word n-grams, and constitutes a genuine privacy threat to anonymous authors. Second, when profiling is largely based on lexical content, DNNs provide the most useful classifier architectures; but the results are more dependent on semantic content and hence less informative about writing style as such. Profiling can be made more sensitive to style rather than content via feature engineering, such as the separation of function words from lexical content. Text representation formats that distinguish between argument structure and grammatical features could be of use here as well (Section 2.2.3; Chapter 4). The comparison of paraphrase choices has also been adopted for stylometry [241], and allows the use of lexical words as features without making the classification overly dependent on semantic content. Increased consideration of such theoretically established feature selection is needed to alleviate the problems raised in this section.

2.2.3 Detailed classification based on semantic and grammatical structure

Publication V presents a novel technique for semantic representation of text that allows detailed information retrieval based on argument structure and grammar. The format is called EAT, and its structure and theoretical motivation are presented in Chapter 4. Here, I instead discuss its potential for tasks that can be broadly considered as text classification; albeit of a very different kind than those surveyed in Sections 2.2.1–2.2.2.

It is possible to distinguish between the argument structure of a sentence, and its grammatical features. For example, sentences (1a–d) all share the
same argument structure, but differ in two grammatical features: tense (present/past) and voice (active/passive). Without going into technical detail here (see Chapter 4), the EAT-format explicitly distinguishes between these aspects, as schematically shown in the examples. The word triplet marks the arguments that remain the same in (1a–d), while the two grammatical markers vary.

(1) a. John sees Mary
   \(<\text{PRESENT, ACTIVE, see, John, Mary}>\)

   b. John saw Mary
   \(<\text{PAST, ACTIVE, see, John, Mary}>\)

   c. Mary is seen by John
   \(<\text{PRESENT, PASSIVE, see, John, Mary}>\)

   d. Mary was seen by John
   \(<\text{PAST, PASSIVE, see, John, Mary}>\)

Via representations like EAT, it is thus possible to create categories of sentences based on argument structure and/or grammatical class. All sentences (1a–d) belong to the same argument structure class; (1a) and (1b) belong to the grammatical class of active; (1b) and (1c) to the class of past; etc. In Publication V this was used for creating parallel corpora between sentences that share argument structure but differ in one grammatical feature, as well as grammatical transformation from one class to another (Chapter 4; Sections 4.3.2–4.3.3). In addition, it is worth asking what other possible applications it could have, in particular related to security and privacy considerations.

One problem we observed in both deception and hate speech detection was that classifiers demonstrated high sensitivity to tokens regardless of their context (Section 2.2.1). The most relevant features for deception are nearly all detectable from word unigrams (Table 2.2). Hate speech classifiers’ vulnerability to word appending attacks did not systematically correlate with model complexity, which also indicates that the models were largely sensitive to unigrams. Hence, to make models more sensitive to semantic relations between words, building such information directly into the input features might be beneficial. Semantic representation formats like EAT can be adopted for such purposes, and EAT is especially useful for implementation due to its simple structure in comparison to alternative formats (Chapter 4; Section 4.2.4).

From a security perspective, the flip-side of such application is the use of detailed semantic information for censorship or the persecution of dissidents. While current techniques are still mostly based around the detection of specific words or their close correlates (based on e.g. embedding distance:
see Chapter 4; Section 4.1.2), combining such measures with more detailed semantic analysis has potential for not only detecting what issues people are talking about, but what stances they take on them. Hence, the more readily available techniques for semantic representation become, the more likely it is that they will also be adopted by institutions that have an interest in suppressing speech.

In author profiling, a major problem we observed was that commonly used features are not reliably stylistic, as opposed to correlating with semantic content (Section 2.2.2). One possible way to alleviate this problem would be more rigorous feature engineering via representations like EAT, which draw an explicit distinction between grammar and argument structure. The high-level grammatical features recognized in EAT are also readily human-interpretable, as opposed to low-level features like character n-gram frequencies. While such low-level features have been successful in stylometry, it is challenging to understand their correlation with features that humans readers would focus on when assessing writing style. Like content classification, stylometry should take model and feature interpretation more into account in future research, as opposed to only relying on basic performance metrics like accuracy or F1-score. Deriving input features from high-level human-readable representations like EAT would facilitate this line of inquiry.

2.3 Summary

In this chapter I have reviewed the main results from Publication I–Publication V on text classification across deception detection, hate speech detection, author profiling, and semantic/grammatical classification. In deception detection, a negative conclusion was drawn: deception might correlate with various (mostly lexical) linguistic choices in certain contexts, but not across contexts. Hence, while NLP might be of use as a part of detecting certain types of deception, there is no evidence of deception as such leaving any sui generis linguistic marks. Instead, alternative techniques based on e.g. semantic similarity metrics or user behavior are more likely to yield useful results on data related to deceptive conduct.

The comparative results on hate speech detection in both Publication II and Publication IV illustrate the importance of training data variability as opposed to model architecture in determining classification success. In small datasets, the most relevant features seem to be sufficiently simple to allow the use of traditional ML techniques instead of requiring complex DNNs. The former also have the benefit of increasing the human-interpretabiliy of model performance. However, our results also indicate that the classifiers get caught on simpler features than those that are actually relevant for hate speech from the perspective of a human reader.
Author profiling by writing style constitutes a genuine privacy threat; although the separation of style from semantic content has not been sufficiently established. Lower-level features like character n-grams and (function) words have displayed more success than more abstract features, despite the latter having a stronger connection to style in human interpretation. Future research should aim at a better understanding of why certain features are most effective in author profiling, and develop more robust methods for separating between style and semantic content – to the extent that this is theoretically possible.

Beyond using characters or (sub)words, deriving input features from a detailed human-readable representation format like EAT can improve the interpretability of text classification across all tasks discussed in this chapter, and allow more detailed classification based on argument structure or grammatical class. While not realistic as a replacement of prior techniques, such an approach is likely to be an useful addition especially in tasks where explaining model performance remains difficult. Complementing the benefits for detecting adversarial text, classification of semantic content and authorship also brings about increased opportunities for security and privacy breaches like censorship and author persecution.
3. Automatic text transformation

This chapter focuses on the second main theme of this dissertation: generating novel texts from original input texts in some way that is systematically linked with properties of the input. I call this text transformation, and include a broad range of techniques under this umbrella term. These range from simple changes applied directly to the input to conditional text generation via encoder-decoder networks.

Section 3.1 discusses the main types of text transformation schemes applied in NLP, divided between rule-based techniques and encoder-decoder techniques. The latter constitute the main contemporary DNN-based framework for text transformation. One of the main topics of this chapter is to evaluate the benefits and drawbacks of these two methodologies across a variety of tasks presented in Publication I–Publication V. I argue that the benefits of DNN- and rule-based transformation techniques are largely complementary, and therefore neither can fully replace the other. The techniques can also be combined, as in Publication V (Section 3.2.3).

Publication I reviews prior approaches to writing style transfer. Our own applications of text transformation range from model evasion (Publication II, Publication III) through data augmentation (Publication IV) to more generic NLP tasks involving detailed control of grammatical features and/or semantic content (Publication V). Sections 3.2.1–3.2.3 review our main results, with a particular focus on the comparison between rule-based techniques and encoder-decoder networks.

3.1 Text transformation techniques

In this section I review the rule-based text transformation techniques and encoder-decoder networks utilized in the dissertation. The range of possibilities is vast, which is why I limit the discussion to approaches used in the novel empirical experiments of Publication II–Publication V. This also covers most style transfer techniques surveyed in Publication I.
3.1.1 Rule-based text transformation techniques

In the present context, I treat rule-based techniques as those that are manually programmed, human-readable, and symbolic. However, it bears emphasis that these properties are at least partly dissociable. For example, ML systems can inductively/abductively construct symbolic rules based on data, as in e.g. grammar inference [79]. Nevertheless, all rule-based techniques used in this dissertation combine all three properties; as opposed to DNN-techniques that share none of them.

Rules being manually programmed also makes them exhibit high human control. This does not necessarily mean that they are fully deterministic, since they can incorporate (pseudo)randomness. Nevertheless, it remains a human-decision where the randomness occurs and what its possible effects can be. For example, a synonym-replacement task (see below) can randomly sample a synonym from a list of candidates; but the fact that such a choice is made at a particular time in the computation is still a hard-coded decision by the human programmer.

Manually programmed rules are also human-readable. Hence, while randomness can prohibit deterministically reverse-engineering the entire process from the output to the input, using rule-based techniques can still allow a high level of human-interpretability for purposes of qualitative evaluation and inference of causal determinants of observed effects. With DNNs, computation inside the network remains uninterpretable, and can at most be partially evaluated with indirect means like attention visualization [16, 305].

Symbolic computation uses operations like replacement, copying, or deletion, which can abstract away from the specific content of the symbols and exhibit abstract relations like variable binding. Even very simple operations like copying an element require variable binding, if the rule is implemented as ‘copy x whatever x is’. The ability to abstract away from specific values has been considered a quintessential property of symbolic computational systems [187]. Another important aspect of symbolic computation is the explicit representation of the objects operated on. For example, in symbolic NLP linguistic objects (such as syntax trees) are represented in some formalism and directly operated on. This crucially differs from DNNs, where no such explicitly defined representations are used, and intermediate structures between input and output emerge from the interplay of numerous hidden node activations.

For present purposes, it thus suffices to characterize rule-based text transformation techniques as those that are based around manually programmed and human-readable symbolic operations that replace original material with something else. I will now review the main types of rule-based techniques applied in the dissertation.
(Sub)word/phrase replacement

One class of text transformation techniques concerns the replacement of subwords, words or multi-word n-grams (phrases). Depending on the task, the replacements can bear different kinds of semantic relations to the original. A common relation is semantic equivalence, which allows the replacement of material without affecting (most aspects of) the original interpretation. Equivalent words are called synonyms and equivalent phrases paraphrases. With other semantic relations like antonymity (contradictory meaning), replacement techniques could also be used for non-paraphrase based tasks like sentiment transfer (e.g. good → bad).

There are two main sources for finding similar tokens for purposes like paraphrasing: knowledge-bases and embedding matrices. The most widely used knowledge-bases for paraphrase replacement are WordNet [201] and Paraphrase Database (PPDB) [97]. WordNet contains information about word senses, which are representations of word meanings that relate to other word senses by various semantic relations, such as synonymity and hyponymy (the subset or if-then relation: e.g. dog is a hyponym of animal).

PPDB is a parallel corpus between plaintext paraphrases, which also contains information about the syntactic context of where the paraphrases appear [97]. It is derived from parallel corpora between English and other languages, used for training machine translation systems. The idea behind PPDB is that two English expressions are likely paraphrases if they are translated in the same way in another language. Subsequent versions of PPDB have incorporated further information about semantic relations between the phrases, similar to that in WordNet [224].

As an alternative to knowledge bases, embedding neighbours can be used as semantically close words for replacement. Embeddings are real-valued vectors derived from a training corpus based on the occurrence contexts of each token in the vocabulary (Chapter 4; Section 4.1.2). Embeddings of tokens with similar occurrence contexts are closer in the embedding space, which allows using vector distance measurements (e.g. cosine similarity) for finding the embedding neighbours of the original token. Word embeddings are most commonly applied, but subword embeddings like BPEmb [123] have increased in popularity due to their better generalizability.

Replacement by embedding neighbours combines DNN-based and rule-based NLP: the training of the embedding matrix is done by a DNN, while the replacement itself is rule-based. Its main benefit in comparison to knowledge bases is flexibility and coverage: all tokens in the vocabulary have embedding neighbours (regardless of absolute distances). On the other hand, embedding similarity is not restrictive in terms of semantic relations: not only synonyms but e.g. antonyms (good / bad) or co-hyponyms (dog / cat) can be close in the embedding space. Whether replacement with embedding neighbours is acceptable depends on the use case.
Text addition
A simple variant of rule-based text transformation is simply adding material to the original input text, without otherwise changing it. While technically trivial, it can be effective at model evasion (Publication II) and improving classification performance when used to augment the original training dataset (Publication IV).

Typo introduction
Automatic techniques can be used to mimic human typos, resulting in spelling variants of the original word that a human reader can reconstruct but that are not recognized as the original word at the relevant computational step (e.g. the embedding layer of a classifier). We applied two algorithms for typo generation, both aiming at maintaining readability.

In Publication II we based our typo generation technique on the finding that character alterations closer to the middle of the word have less impact on readability than alterations toward either edge [251]. We switched the place of two characters, the first randomly drawn from a Gaussian distribution centered around the middle of the word, and the second randomly drawn from a Gaussian distribution centered around the first character. In Publication III we generated typos that mimicked those found in a training corpus for a particular writing style. We built a dictionary from original words to their possible misspellings in the training corpus, using SymSpell\(^1\) for spell-checking. We then chose a random misspelled variant as a paraphrase for the original word.

Grammatical transformation
Controlling grammatical features like tense (past/present) or voice (active/passive) is a challenge for standard rule-based NLP, since many such features are manifested in multiple ways across different sentence types. For example, the active-passive transformation John saw Mary → Mary was seen by John includes changing word order, introducing novel grammatical markers (an auxiliary and a preposition), and changing verb inflection. On the other hand, encoder-decoder techniques have trouble with maintaining tight control of output features, which would be needed for grammatical transformation (see Section 3.1.2).

Prior work has applied rule-based grammatical transformation in limited ways, such as for negation [28] or voice [21]. However, a generic system able to control target sentence grammar has so far been lacking. I present such a system in Publication V. This technique is built around a two-stage process, where the original input is changed into an abstract representation (EAT) given as input to an encoder-decoder network or a rule-based surface realizer for generating the target sentence in English.

\(^1\)https://github.com/wolfgarbe/SymSpell
3.1.2 Encoder-decoder techniques

The other major class of text transformation techniques is based on DNNs that connect two networks in sequence: an encoder and a decoder. In this section I discuss the technical background, and the main types of encoder-decoder networks used in the publications in this dissertation.

**Encoder-decoder RNNs**

Sequence generation can be thought of as sequential classification, where the target class in each iteration is the next token in the generated sequence. When a maximum number of iterations is reached or a specially defined end-of-sequence token (<EOS>) is produced, the generation stops.

Conditional sequence generation via RNNs is similar to sequence classification (Chapter 2; Section 2.1.3), except that instead of the encoder output being given to a classification layer, it functions as the input of another RNN (the decoder) that generates the output sequence. Even though the encoder and decoder are separate networks, it is customary for them to be of the same kind (at least in NLP tasks), e.g. LSTMs or GRUs.

The decoder is a normal RNN, except that in addition to its own prior hidden state(s) and the present input (i.e. its previous output), it also takes the original input encoding into account. If attention is used (see Chapter 2), the decoder additionally looks at the weighted combination of multiple input encoding stages, focusing on the most relevant parts of the original input at each decoding iteration [183]. Figure 3.1 shows the basic structure of a RNN-based encoder-decoder network.
The Transformer architecture discards recurrence in favor of a fully attention-based algorithm, and was specifically developed as an encoder-decoder network [289]. Both the encoder and decoder use a process called self-attention to encode the input sequence in a context-sensitive manner. At each decoding iteration, the decoder first applies self-attention to the sequence it has generated so far, and then additionally attends to the encoder output via encoder-decoder attention. The output of this stage is given to a fully connected layer for generating the next token in the target sequence. Figure 3.2 shows the Transformer pipeline.

In the encoder, the input token is first mapped to a fixed-sized embedding vector together with a marker for its position in the input sequence (positional encoding). The embedding is then mapped to three vectors by multiplying it with a separate weight matrix for each: the key, query, and value. Applying the three matrices to all input embeddings provides the corresponding key matrix \((K)\), query matrix \((Q)\), and value matrix \((V)\), where...
each row is the key/query/value vector of the input token corresponding to the row index position in the input sequence. Using these matrices, the attention score for the inputs is calculated via scaled dot-product attention, shown in Definition 10.

**Definition 10 (Scaled dot-product attention).**

\[
\text{Attention}(Q, K, V) = \sigma(\frac{QK^T}{\sqrt{d_k}})V
\]

where \(Q\) is the query matrix, \(K\) is the key matrix, \(V\) is the value matrix, \(d_k\) is the key dimensionality, and \(\sigma\) is the sigmoid function (Definition 4).

Transformers use multiple independent query, key, and value matrices; each initialized randomly and applied separately to the input. The results are then concatenated and put through another weight matrix that results in the final self-attention output. This process is called multi-head self-attention. Its result is then put through a feed-forward network. Both the encoder and decoder can contain multiple layers, where the output of the feed-forward layer is the input to another instance of self-attention.

To better track the gradient during training, Transformers additionally use residual connections, where the original input is added to the output, and layer normalization [13] is applied to the result for further improving training speed. Residual connections are applied at every layer of the encoder/decoder, both for self-attention and feed-forward connections.

The decoder starts from a special start-of-sequence token (\(<\text{SOS}>\)), and then uses its own output as input with the same kind of self-attention architecture as in the encoder. It then uses the output of this contextual input encoding as the query matrix for key and value matrices derived from the encoder output, at the encoder-decoder attention stage. The decoder output is finally put through a linear fully-connected network and the softmax function, which yields a probability distribution over all possible target tokens. The token with the highest probability is chosen as the next token in the output sequence, and functions as the last input in further iterations of the decoder until the \(<\text{EOS}>\) token is reached.

**Neural machine translation**

The most prevalent NLP task for sequence-to-sequence mapping is translation from one language to another. However, there is nothing in the task itself that requires the inputs and outputs to be taken from different languages. I use the term neural machine translation (NMT) for the technical framework, not the task itself. Hence, e.g. an autoencoder can use NMT to map an input text back to itself. An NMT network is trained on a parallel corpus between source and target text pairs. The network takes the source text as input and is trained to produce the target text. LSTMs with attention had state-of-the-art performance for many years [183, 298], but recently Transformers have obtained superior results [289].
NMT is tailored for settings where large parallel corpora are available for training. Such resources are unavailable for many text transformation tasks. For example, writing style transfer (Publication I; Publication III) or grammatical transformation (Publication III; Publication V) generally cannot rely on pre-existing parallel corpora. Hence, while NMT is the basis of many techniques evaluated in this dissertation, additional means are often needed to avoid this requirement.

Generative adversarial networks
A generative adversarial network (GAN) [104] consists of two parts: a generator and a classifier called the discriminator. The discriminator is trained to classify generator output correctly, and the generator is conversely trained to deceive the discriminator. GANs can be used for style transfer, where the discriminator is trained to classify outputs based on different styles. Most often this has been applied to image data [131, 281, 291], but can also be used for language [271, 272].

Language models
A language model (LM) yields a probability distribution over the next possible tokens, given the previous tokens in the sequence generated so far. In encoder-decoder networks, decoders are LMs that additionally base their output on the context vector(s) generated by the encoder from the original input. LMs can also be prompted by giving them an input sequence that they then use to predict the following sequence.

A benefit in using pre-trained LMs – such as GPT-2/3 [43, 247] – is that they incorporate a vocabulary and grammatical information from a large training set. They can then be prompted on domain-specific input, which should increase the likelihood of thematically similar material in the generated output. Fine-tuning is also possible by training the LM (or only a part of it, such as its final layers) on a smaller domain-specific dataset.

3.2 Thematic content

Following the division of thematic topics in Table 1.1, I review the main results from Publication I–Publication V related to text transformation within each topic. I address the following questions:

- **Adversarial text:** can text transformation be used to (i) evade hate speech detection; or (ii) improve it via data augmentation?

- **Author profiling:** can writing style transfer be used to evade state-of-the-art author profilers, and what are its effects on semantic retention?
• **Semantic representation:** how to conduct effective *grammatical transformation* via an abstract sentence representation that separates semantic argument structure from grammatical features?

All results are attained from our own empirical research, with the exception of those from Publication I, which surveys prior techniques on writing style transfer for evading author deanonymization. Section 3.2.1 discusses the use of text transformation in hate speech detection, Section 3.2.2 writing style transfer, and Section 3.2.3 grammatical transformation.

### 3.2.1 Adversarial text: evading or improving hate speech detection

This section discusses the use of text transformation for evading hate speech detection, as well as improving classification performance on very small training datasets using data augmentation.

*Evading hate speech detection*

In Publication II we used three variants of text transformation rules for creating *adversarial examples* for hate speech classifiers (for discussion of the classifier architectures, see Chapter 2; Section 2.2.1). Each category contained two subtypes, resulting in six transformation techniques overall. These are listed below.

- **word-internal changes**
  1. typos (Section 3.1.1)
  2. leetspeak [226]

- **word boundary changes**
  1. adding whitespace
  2. removing whitespace

- **word appending**
  1. unrelated words
  2. words from the *non-hate* class of the classifier training set

No technique relied on either white- or black-box access to the classifiers. Adding *non-hate* class words from the classifier training set required access to this training set, but the rest of the techniques did not. Hence, five out of these six techniques are available for the adversary *without any access* to the targeted classifier or its training set.

Unsurprisingly, word-internal changes and word-boundary changes were more detrimental to classifiers that used words as input tokens. This is because they break word identities, typically resulting in unknown tokens.
When the training set is highly imbalanced, the unknown token tends to become associated with the majority class, which was the non-hate class in all experiments here.

Mitigating word changes by adversarial training was possible to a degree, but this was more challenging for word boundary changes. In word boundary addition, the combinatorial options are diverse \((n - 1)\) options for a \(n\)-character word), so the likelihood of any particular word part being added to the vocabulary via adversarial training is low especially for longer words. Whitespace deletion on word-based classifiers simply resulted in the whole input document becoming a single (unknown) token, so adversarial training was practically useless.

In stark contrast, with character-based classifiers the effect of word boundary changes was far less, and it was possible to completely undo this negative effect by removing all whitespaces at data pre-processing for both the training and test phases. Given that character-based classifiers also performed similarly to word-based ones at the original tests (Chapter 2, Section 2.2.1), using them was the most effective method to increase model robustness against the attacks without deteriorating original performance.

However, text addition attacks were effective against both word- and character-based models, and cannot be mitigated simply by data pre-processing. They are based on a fundamental problem with using classifier models for detection tasks, which should be indifferent to the presence of unrelated material but fail to be. This attack is similar to prior word appending techniques used against spam filters [180, 316], and illustrates that the problem has not been mitigated by further developments in ML.

Improving hate speech detection via data augmentation
For Publication IV we performed a thorough comparative evaluation across eight data augmentation techniques for hate speech detection.\(^2\) Classifiers used in these experiments are discussed in Chapter 2; Section 2.2.1. Here I instead focus on the augmentation techniques. Table 3.1 reports each technique, its brief description, and its effect on classification performance across the four classifiers experimented on.

We conducted the experiments on Kaggle’s Toxic Comment Classification Challenge dataset.\(^3\) We chose threat as the subtype of hate speech to focus on, and conducted binary classification between it and the remaining documents. The results reported here are from these experiments, but we also applied the best-performing augmentation techniques on another hate speech class (identity hate) with highly similar results. Generically, I refer to the two classes as the minority and majority class, where the minority class consists of hate speech.

\(^2\)Publication IV uses the term “toxic language”, which is equivalent to “hate speech” in the present context.

\(^3\)https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
Table 3.1. Data augmentation techniques used in Publication IV, with their minimum and maximum effects on classification performance on the four classifiers (F1-score). *Char-LR* = character-based LR; *Word-LR* = word-based LR; *CNN* = word-based CNN; *BERT* = fine-tuned pre-trained BERT [81]

<table>
<thead>
<tr>
<th>Technique</th>
<th>Description</th>
<th>Effect on F1-score (macro-averaged)</th>
<th>Min. (classifier)</th>
<th>Max. (classifier)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>Oversampling the minority class</td>
<td></td>
<td>−0.01 (Word-LR)</td>
<td>+0.20 (BERT)</td>
</tr>
<tr>
<td>EDA</td>
<td>Prior technique [293]</td>
<td></td>
<td>−0.01 (CNN)</td>
<td>+0.08 (Char-LR)</td>
</tr>
<tr>
<td>WORDNET</td>
<td>Synonym replacement from WordNet [201]</td>
<td></td>
<td>+0.00 (Word-LR)</td>
<td>+0.18 (BERT)</td>
</tr>
<tr>
<td>PPDB</td>
<td>Paraphrase replacement from PPDB [97]</td>
<td></td>
<td>−0.01 (Word-LR)</td>
<td>+0.20 (BERT)</td>
</tr>
<tr>
<td>GLoVE</td>
<td>Word embedding neighbour replacement</td>
<td></td>
<td>+0.04 (Char-LR)</td>
<td>+0.12 (BERT)</td>
</tr>
<tr>
<td>BPEmb</td>
<td>Subword embedding neighbour replacement</td>
<td></td>
<td>+0.08 (CNN)</td>
<td>+0.13 (Char-LR)</td>
</tr>
<tr>
<td>ADD</td>
<td>Adding random majority class sentences</td>
<td></td>
<td>+0.01 (Word-LR)</td>
<td>+0.21 (BERT)</td>
</tr>
<tr>
<td>GPT-2</td>
<td>Generating novel hate speech sentences with GPT-2 [247]</td>
<td></td>
<td>+0.12 (BERT)</td>
<td>+0.16 (Char-LR)</td>
</tr>
<tr>
<td>AB</td>
<td>ADD + BPEmb</td>
<td></td>
<td>+0.08 (Word-LR)</td>
<td>+0.19 (BERT)</td>
</tr>
<tr>
<td>ABG</td>
<td>ADD + BPEmb + GPT-2</td>
<td></td>
<td>+0.13 (CNN)</td>
<td>+0.19 (BERT)</td>
</tr>
</tbody>
</table>

Augmentation techniques ranged from simple oversampling of the minority class to generating completely novel sentences. Aside of simple oversampling, we divide them to three kinds, reviewed below.

**Sub(word) substitutions.** We replaced words with others based on two kinds of semantic criteria: synonymity/equivalence in pre-existing semantic knowledge bases, and embedding neighbourhood. With two variants of each, this resulted in four original (sub)word substitution techniques. We additionally applied a WordNet-based baseline from prior research [293].

The performance of (sub)word substitution techniques correlated positively with the availability of substitutions, embeddings being more readily available than paraphrases from a knowledge-base. Hence, the best indicator for classification performance across these experiments was how much novelty and variation the augmentations were able to generate. In contrast, the retainment of original semantic content was not particularly relevant here, as embedding neighbour replacement could often enact even drastic (and sometimes ungrammatical) changes in lexical content, based on our manual evaluation on example transformations.

However, an important exception to the generalisation above is the performance of BERT, which resulted in the best classification performance with appropriate augmentation (Chapter 2; Section 2.2.1), in stark contrast to completely failing when trained on seed data alone. However, BERT differed from all other classifiers in terms of which augmentations were the most effective. Compared to them, BERT benefitted much more from simple oversampling. Furthermore, knowledge-base substitutions were more beneficial for BERT than embedding neighbour substitutions. It thus seems that the general pattern of more elaborate changes resulting in better classification improvement was reversed for BERT. As discussed in Chapter 2, this could be due to prior information about word similarities...
being already incorporated in BERT. Since BERT can already use information derived from its large pre-training dataset, it has less need for arbitrarily extending the training vocabulary.

**Majority class sentence addition.** As discussed in relation to evading hate speech detection, adding non-hateful material to original hate speech should not change classification, but can do so unless this is controlled for. One way to mitigate it is adding majority class material to the training set, so that the majority class features become more neutral, i.e. less indicative of the majority class. This should make the classifier more desensitized to irrelevant material, and hence place focus on the most relevant features for the minority class. In Publication IV we used this technique for data augmentation by adding random majority class sentences to original minority class training documents, without changing them otherwise.

As expected, majority class sentence addition was beneficial across all classifiers. The most drastic increase took place in BERT, which provided both the highest (macro-averaged) F1-score (0.71) and the highest F1-score increase from the classifier trained on seed data alone (+0.21). I interpret this result as the combinatorial effect of the increased focus on relevant features during the fine-tuning, and BERT's prior lexical information about (sub)word similarities that allows it to generalize results to tokens beyond those in the training set. Other classifiers benefitted from the first of these aspects but lacked the second, resulting in more modest performance improvement via majority class sentence addition alone.

**Conditional sentence generation by GPT-2.** The most extensive augmentation technique was the generation of completely novel minority class sentences by the Transformer-based LM GPT-2 [247]. We first briefly fine-tuned GPT-2 on the minority class training documents, and then prompted the fine-tuned GPT-2 on each of these original documents.

While BERT benefitted from GPT-2 augmentation, less drastic techniques were often more useful for BERT (see above). In contrast, all other techniques always achieved the highest F1-scores with GPT-2 augmentation. This is well in line with the hypothesis put forward above: expanding the vocabulary was the most important factor for all classifiers except BERT, which already has a very large pre-trained embedding matrix.

Large pre-trained DNN-based LMs like GPT-2 excel at creating synthetic examples with high vocabulary novelty. Notably, this property often stands in contrast to semantic retention of original content, which is maximized by techniques like synonym replacement. Based on Publication IV, vocabulary novelty is more important for data augmentation.

**Best-performing augmentation techniques combined.** The three augmentation techniques with the highest overall performance were sub-word embedding neighbour replacement, majority class sentence addition, and novel sentence generation by GPT-2. We combined all three together by applying each separately to 1/3 of the seed training data. This resulted
in 0.70 F1-score for character-based LR, and 0.69 for all other classifiers. These were the highest scores for all classifiers except BERT.

We also only combined majority class sentence addition with subword embedding neighbour replacement in a 50% – 50% division, to minimize computational load. This combination still achieved 0.69 F1-score with BERT, and 0.62 – 0.68 with the rest. Hence, even without requiring a GPU or extensive memory resources, we were able to closely approximate BERT’s best performance on the character-based LR with appropriate augmentation. These results are well in line with the general observation that the variability and coverage of training data are the most important aspects of classification performance in hate speech detection, as opposed to model architecture (see also Chapter 2, Section 2.2.1).

3.2.2 Evading author profiling via writing style transfer

While there are many possible purposes for writing style transfer (e.g. automatic text simplification [207, 273, 299]), this dissertation focuses on model evasion for evading author profiling. Publication I reviews prior literature on simple techniques: machine translation (MT) trained on style-specific parallel corpora, rule-based paraphrasing, and iterative language translation (ILT). The first of these techniques is obviously insufficient for tackling the problem in any general sense, as the few stylistic parallel corpora that exist are specific only to those particular styles [299, 302].

Prior techniques

A simple out-of-the-box approach to writing style transfer is ILT, where a pre-existing MT system is used to translate the original input across one or more intermediate languages back to the original language (English in all experiments discussed here). Publication I reviews prior studies on ILT across different languages and iteration counts [11, 39, 48, 77, 145, 184, 272]. No overarching consensus can be derived, as they differ in datasets, test hyperparameters, and results. However, some fundamental shortcomings are worth noting. First, ILT is unable to conduct targeted style transfer, since it will only represent the style embodied in the MT system, regardless of how far it is from the original text. Second, the more the original text is changed, the less semantically reliable the result will be. Finally, success in style transfer via ILT seems to be inversely correlated with the success of MT itself. Most ILT studies reviewed in Publication I used Google Translate prior to its update to NMT in 2016. The exception to this [272] failed at style transfer using ILT with Google’s LSTM-based NMT system [298]. Hence, the improvement of MT systems is likely to make them less useful for style transfer via ILT.

4I treat author deanonymization as one type of author profiling (Chapter 2; Section 2.2.2).
Rule-based writing style transfer has generally used synonym/paraphrase replacement from knowledge bases [54, 141, 147, 148, 186] or among word embedding neighbours [185, 254], sometimes with additional manually programmed grammatical transformations related to e.g. punctuation [54, 141, 147]. While they allow a high degree of user-control of the output compared to alternative techniques like ILT, their range of application is typically much smaller. However, in Publication III we present a novel rule-based writing style transfer technique, which uses a combinatorial approach for producing large numbers of paraphrase candidates, and has state-of-the-art performance in a large variety of style transfer tasks (see below).

The main approach in recent style transfer research has been to use encoder-decoder networks, where the encoder is trained to abstract away from the source style and the decoder is trained to generate text in the target style [95, 176, 240, 271, 272, 301, 304, 310]. In Publication III we evaluated the performance of three such techniques in multiple experiment settings. Our overall finding was that none of these managed to combine style transfer with semantic retainment, i.e. the maintenance of the original interpretation of the sentence. When the original sentence was changed, it most often altered the meaning in a detrimental way, making the result unusable for practical application. I briefly review the three encoder-decoder style transfer techniques below.

The cross-aligned autoencoder (CAE) [271] method is based on training a style-specific autoencoder that first aligns the original inputs to a unified encoding distribution with an algorithm called cross-alignment. It then trains a GAN to produce a text close to the original but with style-specific transformations that fool a stylometric classifier trained simultaneously. While CAE sometimes succeeded in style transfer (although its performance varied a lot across datasets), it completely failed at semantic retainment. This illustrates a major challenge in the use of DNNs for tasks that require control of high-level target text properties (like semantic content): it is impossible to verify directly whether or not statistical techniques like cross-alignment are actually able to retain the relevant features, due to the unreadable nature of encodings. The lack of semantic retainment on either manual or automatic metrics in Publication III’s evaluation of CAE indicate that cross-alignment cannot be trusted at this.

The second GAN-based style transfer system – Author Attribute Anonymity by Adversarial Training of Neural Machine Translation (A\(^4\)NT) [272] – first trains a standard autoencoder and a classifier, and then fine-tunes both in a separate GAN-training phase. It additionally enforces proximity to the original text during the GAN-training via a reconstruction loss. Consequently, in contrast to CAE, A\(^4\)NT sometimes managed to retain most of the original meaning. However, A\(^4\)NT very rarely conducted appropriate paraphrasing, and instead maintained a higher semantic re-
tainment simply by minimizing changes to the original. Accordingly, this conservativity resulted in poorer style transfer performance than achieved by the other techniques evaluated.

The third encoder-decoder baseline technique was back-translation (BT) with style-specific decoding [240]. Here, the original English input is first translated to French by a standard pre-trained NMT network. The French translation is then encoded with the French encoder, and two separate style-specific French-to-English decoders are trained. This technique resembles GANs in conducting style-specific training with an external style classifier, but it lacks the dynamic aspect of fitting the decoder and classifier to each other during training. It is essentially a more complex variant of ILT, incorporating the style-specific decoding that ILT lacks (see above). While technically interesting, BT was even less successful than CAE at semantic retainment, and was therefore unusable for style transfer without destroying the original semantic content.

Across all three encoder-decoder style transfer techniques evaluated in Publication III, we concluded that they were unable to paraphrase the original input; instead either avoiding changes altogether (A⁴NT in some datasets) or enacting semantic changes that removed original content (CAE and BT in all datasets; A⁴NT in some datasets). The precise reasons for this are difficult to evaluate directly, but overall it seems that none of the techniques aiming at neutralizing stylistic features of the input (cross-alignment in CAE, back-translation in BT, reconstruction loss in A⁴NT’s GAN-training) managed to do this without altering semantically relevant material as well. Hence, it seems that DNN-based style transfer is difficult to attain, if retaining original semantic content is required.

It is worth noting that what is called “style transfer” in contemporary NLP has often focused on sentiment transfer between positive and negative texts (e.g. reviews) [95, 172, 181, 271, 301, 304]. Unlike writing style transfer in the strict sense, sentiment transfer allows more changes to the original content. The same is true of related tasks like the transfer of political leaning in e.g. Republican-Democrat classification [240]. As these tasks involve large semantic alterations, I do not consider them as style transfer proper in the present context.

ParChoice

Our ParChoice technique is presented in Publication III, and combines multiple rule-based paraphrasing algorithms: simple manually programmed rules, grammatical transformations (Section 3.2.3), paraphrase replacement from WordNet [201] and PPDB [97, 224], and typo injection (Section 3.1.1). First, we applied random combinations of each available paraphrase to the original input, resulting in a large number of candidate paraphrases. From these candidates, we then selected the one that was best able to evade the surrogate author profiler. This modular distinction between
Figure 3.3. ParChoice pipeline (from Publication III).

*paraphrase generation* and *paraphrase selection* makes it possible to use the same paraphrase generation across different settings, with paraphrase selection schemes tailored for each task. This allows ParChoice to be used flexibly across both two-class and multi-class datasets, for both style transfer (changing the original class) and style imitation (targeting a particular class). Figure 3.3 shows the ParChoice pipeline.

In addition to the three encoder-decoder baselines (see above), we compared ParChoice to two rule-based techniques from prior research. The PAN2016 technique [141] is similar to ParChoice in incorporating multiple rule-based methods, but is less semantically restrictive, has less paraphrasing modules, and has a less versatile paraphrase selection scheme. Mutant-X [185] uses word embedding neighbour replacement as the only paraphrasing technique, and incorporates this to a *genetic algorithm* that selects the best-performing candidate as the next input in each iteration, based on black-box access to the targeted author profiler.

Unlike the encoder-decoder techniques, ParChoice was able to retain semantic content in the majority of cases evaluated manually, and had significantly better scores in automatic evaluation of semantic retainment. It also exhibited superior style transfer performance to the only encoder-decoder technique that was ever able to retain semantics to a non-negligible degree (A4NT). Furthermore, ParChoice markedly improved semantic retainment performance from both rule-based baselines. Mutant-X had higher style transfer success than ParChoice alone, but a combination of Mutant-X and ParChoice outperformed Mutant-X in both semantic retainment and style transfer.

Of the rule-based approaches compared, Mutant-X had the lowest semantic retainment. This indicates that using word embedding neighbours for replacement did not reliably result in appropriate paraphrases. On the other hand – most likely due to their wider vocabulary range – word embeddings allowed Mutant-X to find optimal candidates for style transfer even when PAN2016 or ParChoice could not. Therefore, their application in tasks like style transfer should be *minimized*, and only done if more semantically restrictive alternatives (like ParChoice) have first been exhausted. This was implemented in the combination of ParChoice and Mutant-X, which had the highest performance overall.
3.2.3 Grammatical transformation

In Publication V and Chapter 4 I discuss our EAT format for text representation. Since EAT was used for text transformation (among other tasks), in this section I describe those applications without going into detail on the structure of EAT. As explained more thoroughly in Chapter 4, EAT is a format for representing semantic and grammatical content in a manner that separates them into different parts in a sequence. As EAT is fully human-readable, both grammatical and lexical features can be freely altered by simple replacement rules. In the case of grammatical features such transformations are maximally simple: one bit of information encoded as 0 or 1 in a dedicated position.

We trained a NMT network to generate English from EAT by applying a vectorized variant of EAT as input to an encoder LSTM, and training a decoder LSTM to generate the original English sentence from which the EAT was derived (via syntactic parsing). Essentially, this architecture is like that of standard NMT networks, except that the encoder has no embedding layer, and instead receives a pre-constructed vector sequence as the direct input. This vector directly encodes the EAT by using Boolean features for grammatical properties and pre-trained word embeddings for words. We call the whole encoder-decoder network EAT2seq.

Once the EAT2seq model is trained to reproduce English from EAT, it is possible to provide it transformed EAT-inputs, by first constructing the EAT from the original (parsed) sentence, and subsequently applying the transformations directly on the EAT prior to giving it as input to the encoder. Crucially, any (grammatically allowed) transformation is possible to conduct with a single network trained once, with no task-specific training. In Publication V we applied grammatical transformations between 14 classes with a single EAT2seq network. As an alternative, we also used a rule-based surface realization system for translating EAT to English with corresponding grammatical transformations (Chapter 4; Section 4.3.3).

A major benefit of EAT2seq compared to purely DNN-based text transformation techniques (Section 3.1.2) is that EAT2seq uses NMT for text generation but not for the transformation itself, which is instead conducted via symbolic rules on the EAT-input. In Section 3.2.2 I argued that while DNNs can be used for transformations that enact varied and large-scale changes (as in sentiment transfer), their application for transformations that require detailed control of output features is problematic. EAT2seq uses NMT where it works the best: mapping one type of representation (EAT) to another (English). However, EAT2seq uses another technique for the transformations themselves, which are more challenging for DNN-based approaches. This way, EAT2seq can be thought of as a hybrid technique that maximizes the benefits of rule-based and DNN-based methods within a single (but modular) architecture.
3.3 Summary

In this chapter I reviewed the text transformation techniques evaluated in this dissertation, and the main empirical findings observed. Continuing the theme of comparing DNNs to rule-based NLP, here we can especially clearly see their complementary benefits and drawbacks. This brings to question the notion that DNNs could replace rule-based NLP entirely, even if sufficient computational resources and training data were available. Instead, my recommendations on which techniques are the most useful depend crucially on the task.

DNNs excel at two kinds of tasks. The first is what can be called supervised translation, where a parallel corpus is used to train an NMT network to perform sequence-to-sequence mapping. Current state-of-the-art results have been achieved with DNNs like LSTMs [183, 298] and – more recently – Transformers [289]. Hence, if large parallel corpora between input and output text classes (languages, styles etc.) are available, DNN-based NMT will most likely yield the best performance available.

The challenge in supervised translation is that large parallel corpora are not always available, especially between tasks tailored for particular purposes: style transfer to a specific target style, grammatical transformation to a particular direction, etc. Another shortcoming of supervised translation is its uni-directionality: a network trained to translate between one type of input and one type of output is not able to reverse this translation or target any other output class without re-training. Especially for versatile tasks like style transfer or grammatical transformation, training separate NMT networks for each input-output class pair is uneconomical.

The second task that DNNs have showed great promise in is the generation of novel text with thematically appropriate but indeterministic output (to a human reader). I call this conditional generation, since the output is only conditioned by the input but not determined by it. LSTM-based techniques have been used to generate e.g. realistic fake reviews [137]. More recently, large pre-trained Transformer LMs like GPT-2/3 [43, 247] have demonstrated the ability to generate texts that maintain thematic coherence even across long distances.

The purpose of conditional generation is partly opposite to that of supervised translation. In supervised translation the NMT network is meant to produce the correct translation of the input; whereas in conditional generation there is no single correct output, but instead an open-ended set of thematically appropriate outputs. Indeed, it is precisely because the output cannot be determined from the input that LMs like GPT-2 can seem genuinely “creative” from the perspective of a human reader. In contrast to supervised translation, the main benefits of conditional generation are thus the novelty and variability of the output. Publication IV demonstrated these to be highly important for data augmentation.
Roughly, DNNs seem to perform well when the output sequence is either *strictly determined by the input sequence* (supervised translation), or when it should be *sufficiently random to be perceived as genuinely novel* (conditional generation). Importantly, many tasks fall between these extremes: here, the output is determined by the *input plus some transformation parameter(s)*. In fact, most text transformation falls in this camp. For example, in style transfer (proper) only semantically irrelevant features should be changed; and in grammatical transformation everything should be retained except for the relevant grammatical features. Our results illustrate the limits of encoder-decoder networks in these kinds of tasks, and combining their benefits with those of rule-based NLP continues to be relevant here.
4. Text representation

The final main topic of this dissertation goes beyond immediately security-related applications toward a more fundamental question for NLP in general. A connection to prior topics is still maintained, as both classification and transformation of text are discussed in this part as well. I already invoked the EAT format in relation to semantic/grammatical text classification (Chapter 2; Section 2.2.3) and grammatical transformation (Chapter 3; Section 3.2.3); and this chapter focuses almost entirely on Publication V, which proposes EAT. The main case put forward is that a major reduction of semantic roles is possible both theoretically and in practice, which opens up new possibilities for the use of semantic representations beyond those available in NLP applications so far.

Section 4.1 is a brief survey of prior approaches to semantic representation in NLP. The overarching theme of comparing DNNs to rule-based NLP brings about especially clear differences in this domain. Continuing the line of argument from the prior chapters, I argue the benefits and drawbacks of these frameworks are largely complementary, which motivates using both in combination when possible. Section 4.2 then presents the EAT format from Publication V, by first reviewing its main theoretical motivations (Sections 4.2.1–4.2.3), and then providing the definition along with its representation as a vector sequence (Section 4.2.4).

Section 4.3 discusses the experiments conducted for Publication V: comparing a reduced variant of a prior representation format to the original (Section 4.3.1), parallel corpus construction between grammatical classes (Section 4.3.2), and text reconstruction or transformation (Section 4.3.3). The last of these was already partly covered in Chapter 3 (Section 3.2.3). There the focus was on the hybrid nature of EAT2seq as a text transformation tool, whereas here I focus specifically on why EAT makes this task more feasible than alternative approaches to semantic representation.
4.1 Techniques for representing semantic content

In a broad sense, all NLP tasks require some form of input text representation. The simplest representation format is bag-of-words (characters), which records only the presence of tokens and nothing else. N-grams record surface-level relations between adjacent tokens directly into the features. One major problem here is that partially overlapping n-grams are not related: for example, bigrams like <I, am> and <you, are> are as far apart as e.g. <I, am> and <this, dog>. More sophisticated representation techniques aim at recording relational information between tokens without removing information about the relata themselves. Section 4.1.1 reviews explicitly defined semantic metalanguages both in terms of theoretical background and existing NLP applications. Section 4.1.2 continues the discussion (already begun in prior chapters) on the possibilities and limitations of data-based learning of semantic representations via DNNs.

An important semantic property of (many) complex linguistic expressions is compositionality. The principle is at least implicitly assumed in standard frameworks of linguistic research (e.g. descriptive grammars), but its first proper formulation has been assigned to Frege [83, 94]. The main point of Frege’s account is that the meaning of a complex expression is a function of the meanings of its constituent expressions and their mode of combination. One immediate problem with this generic definition of compositionality is that the mathematical notion of function is too liberal to make the principle sufficiently informative for purposes of understanding semantics. Intuitively, the idea behind compositionality is that the meaning of a complex expression is not only a function of its constituents, but somehow built from those constituents in a transparent manner. From the perspective of a system that processes expressions (e.g. a human or an NLP application), this means that the system is able to compute the meaning of the complex expression based on knowing its constituents and their mode of combination.

A stronger requirement on compositionality is that the constituents and their mode of combination should be recoverable from the meaning of the complex expression. That is, not only should the meaning of a complex expression be a function of its parts and their mode of combination, but this relation should also be reversible: the complex expression should reveal its constituents and their mode of combination [91, 218]. This has direct relevance for NLP, since I claim that it is precisely where DNNs are likely to fail in comparison to symbolic representation techniques. If correct, this means that available methods for explicitly representing compositional semantics remain symbolic.
4.1.1 Semantic formalisms

In contrast to DNNs, symbolic NLP approaches to semantics typically use formally defined semantic metalanguages that have *direct computational relevance*, such that computational operations make explicit reference to expressions in the metalanguage. This section briefly explains the theoretical background related to such formalisms, and three common NLP applications prior to EAT.

*Logical forms*

The logical form (LF) of a sentence is meant to capture those aspects of its meaning that are relevant for its *truth conditions*. Many aspects of linguistic meaning – understood in a broad sense – are not truth-conditional: connotations, pragmatic considerations, etc. Truth-conditionality has a special relation to the principle of *compositionality* discussed above: truth-conditions are compositional, while non-truth-conditional aspects of meaning are not (at least necessarily). Hence, LFs are specifically tailored for making explicit those aspects of complex expressions by virtue of which their truth-conditions are determined via the truth-conditions of their constituents and their modes of combination.

The most prevalent approach to LFs stems from Frege [94], and was utilized for natural language by Montague [204, 205]. Here, when two expressions are linguistically combined, one of them is semantically interpreted as a *function* that takes the other as an *argument*. Technically, this is typically implemented using a mathematical technique called *lambda-calculus* [65], which allows specifying functions of arbitrary complexity [122, 223].

The Fregean ontology recognizes two atomic semantic types: *entities* (<e>) and *truth-values* (<t>). Every other semantic type is a function between some semantic types, such that saturating the function with an element of the appropriate type yields the target type. For example, a predicative copula-adjective expression like *is alive* can be treated as a function from entities to truth-values based on whether they are alive or not. In (informal) lambda-calculus notation, this can be denoted as \( \lambda x : [x \text{ is alive}] \), and its semantic type as <e, t>. That is, when combined with something \( x \) of type <e>, it will yield a truth value (of type <t>) which is *True* if \( x \) is alive and *False* otherwise. This is illustrated in (2) for the sentence *John is alive*, where *John* is of the type <e>.

\[
(2) \quad \text{John is alive} \quad \lambda x : [x \text{ is alive}] \\
\text{John} \quad <e> \quad <e, t> \\
< t >
\]

The Montagovian approach provides a powerful descriptive framework
for formalizing compositional semantics, and has been used in some NLP applications [252, 253]. However, once expanded to account for a more varied range of expressions, it quickly becomes highly complicated, requiring the postulation of very complex semantic types and/or additional operations like type-shifting [222].1 While the framework itself is able to withstand such complications, especially from an applied perspective the use of complex semantic types becomes difficult to handle.

One way to simplify LFs is to adopt a Neo-Davidsonian account of verb semantics. The classical Fregean analysis of verbs treats them as predicates satisfied by thematic arguments [122, 204]. Here, the verb's valency is an intrinsic feature that determines its number of arguments. An intransitive verb like run takes a single argument, and a transitive verb takes two. In terms of Montagovian semantic types, intransitive verbs are of the type $<e, t>$, and transitive verbs of the type $<e, <e, t>>$, as shown in (3).

![Diagram of LF for John runs and John sees Mary]

An alternative analysis was presented by Davidson, and adds a event variable to each verb [53, 75]. This allows treating e.g. adverbial modifiers as additional predicates over the same event variable, as in (4).

![Diagram of LF for John runs fast]

The Neo-Davidsonian analysis goes away further from the Fregean view, separating thematic arguments from the verb into separate dyadic relations with the event variable [124, 221, 265]. As shown in (5), I adopt the generic terms Agent and Theme for the two thematic arguments of a transitive verb, and Agent for the single thematic argument of an intransitive verb (but see Sections 4.2.3–4.2.4 for discussion on unaccusativity).

---

1In fact, example (2) is in opposition to Montague’s own approach to proper names, which aimed at unifying them with quantified noun phrases, resulting in a far more complex semantic type than $<e>$ [205]. The simpler approach presented here was advocated by Partee [222], but this in turn required her to apply type-shifting operations to proper names in certain contexts. This is one indication of the complexities involved in applying the Montagovian approach beyond simple examples.
(5) a. John runs  
$\exists e [\text{RUN}(e) \land \text{Agent}(e, \text{JOHN})]$  
b. John sees Mary  
$\exists e [\text{SEE}(e) \land \text{Agent}(e, \text{JOHN}) \land \text{Theme}(e, \text{MARY})]$  

Despite making the LFs longer and hence seemingly more complex, Neo-Davidsonian semantics has important simplifying effects on the theory. All verbs, regardless of valency, are now *assimilated* in semantic type: they are simply monadic (single-argument) predications over an event variable. This can be seen by comparing (3) and (5): in the former (Fregean) variant the intransitive and transitive verbs had different semantic types: $<e, t>$ and $<e, <e, t>>$. In the latter (Neo-Davidsonian) variant *both* are of the type $<e, t>$, but in a novel way. Here, the entity position is filled not by the Agent argument but by the event variable.

By splitting thematic roles from the verb’s intrinsic argument slots, Neo-Davidsonian semantics achieves a simple treatment of *transitivity alterations*, where the same verb can appear with or without the Theme argument. It also allows a straight-forward analysis of *passives* where the Agent is omitted.

(6) a. John bakes bread  
$\exists e \exists x [\text{BAKE}(e) \land \text{Agent}(e, \text{JOHN}) \land \text{BREAD}(x) \land \text{Theme}(e, x)]$  
b. John bakes  
$\exists e [\text{BAKE}(e) \land \text{Agent}(e, \text{JOHN})]$  
c. Bread is baked  
$\exists e \exists x [\text{BAKE}(e) \land \text{BREAD}(x) \land \text{Theme}(e, x)]$

From the perspective of NLP, a further benefit of removing argument structure information from the verb’s intrinsic semantic type is that it takes us away from the “lexicalist” requirement of storing such information for each verb explicitly. Resources like FrameNet [17, 262] and PropBank [219] specify the thematic arguments required by specific verbs; but while they are useful, it is better if applications are not completely reliant on them. Semantic representations should be constructed even for verbs that are not present in such knowledge-bases.

In linguistic theory, the main non-lexicalist alternative to finding verbal arguments is to read them directly from the *grammatical context*. This links Neo-Davidsonian semantics to analyses of the syntax-semantics interface where syntax provides a “spine” or “skeleton” onto which arguments are latched [35, 117, 177, 249]. Grammatical context is, of course, relevant for any semantic parsing system. However, in a non-lexicalist system it not only necessary but *sufficient* for attaining the semantic representation. The feasibility of such a system depends on many factors: what kind of a syntactic representation is used, how many semantic roles are required, etc. I discuss the linguistic background in Section 4.2.1.
Discourse Representation Theory

DRT is a semantic framework mainly developed by Kamp [139, 140]. It maintains that semantic interpretation consists in building discourse representation structures (DRSs), to which elements are dynamically added when the discourse unfolds. A DRS consists of a list of discourse referents (DRs), and a list of DRS-conditions, which are predications over DRs. An example DRS is shown in (7), using the standard box notation of DRT and Fregean semantics for the verb. Conditions inside a DRS are interpreted as conjoined within the DRS.

(7) A dog runs

\[ x \text{ DOG}(x), \text{ RUN}(x) \]

DRSs can enter into relations corresponding to propositional connectives in classical logic. In DRT, these are called complex DRS-conditions [139, 140]. DRSs can be sub-DRSs of larger DRSs that express the propositional relations. Complex DRS-conditions can also be used to account for quantification and modal operators. Example (8) is a DRS for a simple conditional clause with two sub-DRSs.

(8) A dog runs if a cat walks

\[ x \text{ CAT}(x), \text{ WALK}(x) \rightarrow y \text{ DOG}(y), \text{ RUN}(y) \]

While DRT was not developed for NLP purposes unlike e.g. AMR or MRS (see below), it provides a relatively simple but powerful semantic representation format for computational application. The most extensive DRT-based semantic parsing tool so far is Boxer [36], which departs from classical DRT by incorporating Neo-Davidsonian event semantics.

Abstract Meaning Representation

AMR [19] is a graph-based representation of variables (nodes), concepts that predicate them (node labels) and semantic relations (edges). It recognizes \( \sim 100 \) semantic relations overall, which come in many kinds: abstract argument slots from PropBank [219], additional semantic roles (e.g. beneficiary or destination), negation, modality, etc.

AMR abstracts away from various types of grammatical information, some of which has semantic impact, such as tense inflection. It also assimilates between some syntactically divergent expressions with logically equivalent interpretations. Using an example from Banerescu et al. [19], the following sentences all have the same AMR analysis: (i) he described her as a genius; (ii) his description of her: genius; and (iii) she was a genius, according to his description. This distinguishes it from MRS (see below), which remains closer to the surface syntax.

Due to AMR's reliance on detailed lexical information (e.g. word frames from Propbank) and its non-transparent link to syntax, creating AMRs
from text is challenging. Small gold standard AMR-datasets have been produced manually, and parsing text into AMR has been done via sequence-to-sequence architectures in NMT-type settings [154]. However, MRS has fared better in corresponding experiment settings [116, 174].

**Minimal Recursion Semantics**

The main motivation behind MRS is to provide a simple semantic representation for optimal use in NLP applications [70, 72]. It has specifically been designed for compatibility with typed feature-structure syntactic parsing frameworks, in particular Head-driven phrase structure grammar (HPSG) [237]. Dependency-MRS (DMRS) is a graph-representation of MRS, which uses nodes for words and edges for semantic relations [71]. Like AMR, DMRS allows linearization via graph-traversal [105, 106].

There are at least three major ways in which MRS maintains more proximity to surface grammar than AMR. First, MRS allows ambiguity when disambiguation is not possible from the syntactic parse alone. This goal importantly differentiates it from standard LF-based analyses, where disambiguating natural language is one of the main tasks. For example, in the sentence *Everybody saw somebody* the universal and existential quantifiers can be interpreted in either order, which is not specified by syntactic structure (at least on the surface-level [190, 191]). This means that the sentence has two distinct LFs, shown in (9a–b) (with Fregean verb semantics for notational brevity).

\[
\begin{align*}
\text{(9)} & \quad \text{a. } & \forall x \exists y \ & \text{SEE}(x, y) \\
& \quad \text{b. } & \exists x \forall y \ & \text{SEE}(x, y)
\end{align*}
\]

In contrast, since the ambiguity is not resolved by syntax alone, the MRS format continues to allow it by only having one representation for such sentences. While AMR also retains some ambiguity with respect to quantification, in MRS this property is an explicit goal (rather than a perceived deficiency as in Banarescu et al.’s [19] discussion on AMR’s limitations).

Second, MRS contains more grammatical information than AMR. It retains e.g. tense inflection, and does not assimilate between syntactically divergent expressions as readily as AMR. This somewhat reduces its ability to assimilate between equivalent sentences, but better maintains its proximity to the syntactic parse and hence improves its applicability for text generation [116].

Third, the semantic roles in MRS are not specified for PropBank frames or any other pre-existing lexical information. On the one hand, this means

\[\text{As Banarescu et al. [19] point out, inflectional information like tense could also be added to AMR. In contrast, the assimilation of syntactically divergent sentences is a foundational feature of AMR that could not be changed without significant alterations to the format.}\]
Table 4.1. Taxonomy between three prior semantic metalanguages used in NLP.  
(Transparency = how well semantics can be mapped from syntax alone.)

<table>
<thead>
<tr>
<th>Number of semantic roles</th>
<th>Transparency High</th>
<th>Transparency Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>MRS</td>
<td>AMR</td>
</tr>
<tr>
<td>Low</td>
<td>DRT</td>
<td></td>
</tr>
</tbody>
</table>

that the roles are less readily semantically interpretable than in AMR. On
the other hand, it simplifies the link between the syntactic parse and MRS.
However, like AMR, MRS still relies on a large number of semantic roles.
For example, the dataset used by Hajdik et al. [116] has 52 roles overall.

Empirically, MRS has demonstrated superior performance in parsing
and text generation than AMR. Hajdik et al. [19] used NMT to translate
DMRS to English, and achieved a much higher BLEU score (75.8) than
prior work using AMR on the same dataset (33.8). Lin and Xue [174]
provide a detailed comparison between MRS and AMR in terms of parsing
and text generation accuracy, and account for MRS’s generally superior
performance on three main grounds: (i) AMR’s higher degree of abstraction
from surface forms, (ii) AMR’s finer-grained classification of named entities,
and (iii) MRS’s semantic roles bearing a closer relation to syntactic roles.

Interim summary: taxonomy of semantic metalanguages in NLP
I briefly reviewed three examples of prominent semantic frameworks used
in NLP: DRT, AMR, and MRS. We can roughly divide these techniques
based on two criteria: number of semantic roles, and transparency of the
syntax-semantics mapping. Table 4.1 summarizes this taxonomy.

DRT is designed to be derivable from the syntactic struture [139, 140],
and in this sense it has the requirement of at least some transparency
(like all semantic parsing). Nevertheless, its structure requires the use of
devices that cannot be based on syntax alone. One of the main goals of DRT
is to account for discourse-semantic factors like anaphor resolution, which
adds information beyond what is available in the syntactic parse alone.
While this can be useful for certain NLP tasks, it adds to the difficulty
of semantic parsing. On the other hand, the number of semantic roles
required in DRT is smaller than those used in AMR or MRS. It is of course
possible to add more semantic roles to a DRT-based representation; but
e.g. Boxer [36] uses only five proto-roles. DRT can therefore be allocated to
the class having a (relatively) non-transparent syntax-semantics mapping
while allowing only few semantic roles.

Like DRT, AMR abstracts away from surface syntax [19]; but also uses
~ 100 semantic roles. I therefore allocate it to the class with a non-
transparent syntax-semantics mapping and a large number of semantic roles. MRS is an attempt at providing a semantic representation only based on information in the syntactic parse [70, 72]. However, MRS still remains complex in structure, using dozens of semantic roles. Hence, I classify it as having a transparent syntax-semantics mapping and a high number of semantic roles.

One cell is missing in Table 4.1: combining a transparent syntax-semantics mapping with only few semantic roles. While this has been the aim of certain lines of research in semantic theory, it has so far not been central in NLP applications. Publication V tackles this problem.

4.1.2 DNN-based representation techniques

Chapters 2–3 have discussed empirical results on DNN-approaches to text classification and transformation from Publication I–Publication V. Their ability to retain semantics across e.g. stylistic changes was deemed unsatisfactory (Chapter 3; Section 3.2.2). However, the discussion so far has not concentrated on how DNNs could represent semantics, and why they might succeed or fail at it. In this section I focus on such questions.

Embeddings

Embeddings are built around another major tradition in the theory of linguistic meaning aside of formal semantics: *distributional semantics*. The basic idea here is that the meaning of a word (or some other meaningful token such as a morpheme or a subword) is based on its relations to other words/tokens in language use. Although not without contenders, distributional semantics has had major influence in linguistics, philosophy of language, and cognitive science [30, 86, 89, 90, 121, 297].

The two most prominent word embedding techniques are *word2vec* [200] and *Global Vectors for Word Representation (GloVe)* [225]. Both produce fixed-size real-valued vectors of tokens in a training corpus based on their co-occurrence patterns with other tokens. GloVe takes the global context into account, whereas word2vec only looks at the words nearby to the target. The two main word2vec algorithms are *contextual bag-of-words (CBOW)* and *skip-gram*. CBOW predicts the target word based on its context (Figure 4.1); and the skip-gram variant goes to the other direction by assigning probabilities from the target word to its neighbours (Figure 4.2). GloVe uses a global co-occurrence matrix across all words as the target. In all techniques, the embedding itself is the final hidden state of the DNN that is used to predict the target word or distribution.

Because embeddings are calculated based on occurrence distributions of tokens in the training corpus, tokens with similar distributions will be close in the embedding space. If the assumption of distributional semantics holds, this should systematically correlate with semantic similarity, as
understood in some theoretically appropriate sense (see below for more discussion). Such proximity is commonly measured by cosine similarity (cos(θ): Definition 11).

**Definition 11 (Cosine similarity).**

\[
\text{cos}(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| ||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

where \(\mathbf{A}\) and \(\mathbf{B}\) are vectors, and \(A_i\) and \(B_i\) their components.

Embeddings are among the most prominent and efficient ways to represent words in NLP. They can be learned automatically from a corpus with minimal pre-processing (tokenization), and neighbourhood in the embedding space often correlates with semantic similarity (but see below). Using subword tokens [158, 270] instead of words further helps the model to deal with unknown words and misspellings [123]. Nevertheless, while the benefits of embeddings are undeniable, it is worth asking what their shortcomings are. Importantly, I argue that their drawbacks are largely the opposite of those of traditional LFs or manually derived lexical knowledge bases, and vice versa.

In Publication III we compared style transfer based on word embedding neighbours [185] to style transfer based on paraphrase replacement from lexical knowledge bases (Chapter 3; Section 3.2.2). Word embeddings were more widely applicable, but less reliable in semantic retainment. We thus recommended the use of lexical knowledge bases as the first option, and embedding neighbours only if the former are insufficient for style transfer, based on task-specific criteria such as evading an author profiler.

Our results on style transfer speak to a fundamental challenge with embeddings: while embedding neighbours exhibit semantic similarity in some sense, this similarity cannot be defined more strictly in terms of e.g. truth-conditional criteria. Embedding neighbours can be e.g. synonyms (car and automobile), co-hyponyms (cat and dog), or antonyms (good and bad). There is some semantic relation between each of these cases; but the embeddings themselves do not show the nature of this relation. In
particular, there is no way of enforcing the relation to be stricter from a logical perspective, such as truth-preserving. Whether this is a significant problem depends entirely on the NLP task the embeddings are used for. Sometimes it is beneficial not to distinguish between different kinds of semantic similarity; but in certain tasks it is vital. In the latter, reliance on embedding neighbourhood as a direct semantic similarity metric is risky.

Another fundamental problem with embeddings is their application to complex meanings. The principle of compositionality states that the meaning of a complex expression is a function of the meanings of its constituents and their mode of combination. While there are many simple ways to combine vectors (e.g. summation, multiplication, averaging, etc.), these do not exhibit compositionality in a sufficient way; lacking in basic requirements like sensitivity to word order.

There have been attempts to incorporate formal semantics directly to vector(matrix/tensor) representations [22, 86, 202, 203]. For example, Baroni et al. [22] distinguish between (Montagovian) types based on the syntactic parse, and represent semantic functions as distributional functions that map embeddings to other embeddings. Since such approaches rely on prior (syntactic) information and explicit distinctions between semantic types, I consider them to exemplify hybrid NLP instead of purely data-based techniques. While interest in incorporating semantic functions to distributional representations has not been central in more recent NLP, I believe it deserves more attention in future work.

**Encodings as input representations**

As an alternative to building complex meanings directly from embeddings via vector combination, the main DNN-based technique for representing complex linguistic sequences is to use encodings. The process of producing an encoding from an input sequence via RNNs (e.g. LSTMs or GRUs) was discussed in Chapter 2; Section 2.1.3. The encoding is the final hidden state of the RNN that has gone through the entire input sequence. Encodings are sensitive not only to the presence of input tokens but their positions, which makes them more appropriate for distinguishing between meanings of sentences based on word order.

However, state-of-the-art techniques in sequence modelling complicate the matter, since they rely on multiple input encodings instead of only one. The vanishing gradient problem in vanilla RNNs was the consequence of latter parts of the input having more influence than earlier parts. While LSTMs [126] and GRUs [64] provided help with this problem, the most major improvement was the addition of attention first to LSTMs [183], and subsequently basing the entire network only on the hierarchical use of attention layers [289, 305] (Chapter 2, Section 2.1.3). In attention-LSTMs the final encoding has much less influence than before; while Transformers [289] use attention over (the encodings of) all input tokens (Chapter 3,
Section 3.1.2). This more dynamic and distributed approach makes it more challenging to individuate any singular input representation.

BERT [81] includes an artificial addition for producing a single encoding of the whole input string. Each input is concatenated to a [CLS] token in the initial position, and the encoding corresponding to it serves as the input for tasks that rely on singular vectors (such as classification). This [CLS]-encoding can be considered as the closest correlate to the encoding of the whole sentence with a Transformer network like BERT. There are also other alternatives that are based on pooling all input token encodings: e.g. averaging or taking their maximum values [255]. The performance of sentence encoders is obviously highly dependent on the task they are trained on. A common strategy has been to train the network to compare sentence pairs for semantic relations on human-labeled data, as presented in corpora like SNLI [37, 55, 69, 255]. In contrast, pooled sentence encodings on BERT or RoBERTa [175] have not exhibited good performance when applied to sentence similarity comparison tasks directly without such supervised training [255].

Do sentence encodings represent compositional semantic structure?
The issue of representing complex sentences brings us back to a dilemma mentioned in the beginning of this section, concerning the characterization of compositionality. The original Fregean definition is simply that the meaning of a complex expression is a function of its constituents and their mode(s) of combination [94]. A stricter notion of compositionality is that the meaning of a complex expression should not only be based on its constituents but contain them, such that the constituents and their mode of combination could be retrievable from the complex meaning alone [91, 218]. This difference can be illustrated with a simple toy example.

Let $L$ be a language that contains all eight sentences derivable from the formula (10).

(10) $A \ B \ C$

where $A \in \{John, Mary\}$, $B \in \{\text{is, 's}\}$, and $C \in \{\text{nice, tall}\}$

Let $Enc_1$ be an encoding function which maps the sentences of $L$ to two-dimensional vectors with binary components, as shown in (11).

(11) $Enc_1 =$

\[
\begin{align*}
(John \ is \ nice, & \ [0, 0]), \quad (John's \ nice, \ [0, 0]), \\
(Mary \ is \ nice, & \ [1, 0]), \quad (Mary's \ nice, \ [1, 0]), \\
(John \ is \ tall, & \ [0, 1]), \quad (John's \ tall, \ [0, 1]), \\
(Mary \ is \ tall, & \ [1, 1]), \quad (Mary's \ tall, \ [1, 1])
\end{align*}
\]

Assuming sentences of $L$ to share the interpretation of their English counterparts, $Enc_1$ is straight-forwardly compositional. The first component of each vector specifies the subject ($John$/$Mary$), and the second component
specifies the adjective predicated over the subject (nice/tall). Furthermore, paraphrases using either the full copula verb (is) or its contracted variant (’s) are assimilated, which makes the resulting vectors abstract away from this aspect of surface grammar. Vector distance also systematically correlates with constituent overlap.

Now, let $\text{Enc}_2$ be another encoding function which maps the sentences of $L$ to real-valued one-dimensional vectors (i.e. numbers), as shown in (12).

\begin{equation}
\text{Enc}_2 = \{(\text{John is nice}, 0.0), (\text{John's nice}, 0.0), (\text{Mary is nice}, 0.3), (\text{Mary's nice}, 0.3), (\text{John is tall}, 0.7), (\text{John's tall}, 0.7), (\text{Mary is tall}, 1.0), (\text{Mary's tall}, 1.0)\}
\end{equation}

$\text{Enc}_2$ shares certain properties with $\text{Enc}_1$: equivalent sentences have the same encoding, and encoding distance correlates with constituent overlap. Both observations could be used to defend the notion that $\text{Enc}_2$ represents the sentence meaning in some sense. However, unlike $\text{Enc}_1$, $\text{Enc}_2$ does not contain components that correspond to constituents of the original sentences. While both $\text{Enc}_1$ and $\text{Enc}_2$ are compositional in one sense (there being a function from syntactic structure to semantic interpretation), only $\text{Enc}_1$ represents semantic structure via the structure of the encoding itself. One way to put this distinction is that $\text{Enc}_1$-encodings are disentangled while $\text{Enc}_2$-encodings are entangled.

Separating different components from complex representations is often required for NLP tasks like information retrieval or inference. Therefore, a crucial question for sentence encodings is whether this is feasible: i.e. whether the encodings are sufficiently disentangled with respect to semantic constituents. As they are not human-readable, this can only be evaluated indirectly. However, as (12) illustrates, even an entangled representation like $\text{Enc}_2$ could succeed in mapping equivalent sentences to the same (or close) encodings, and more highly divergent sentences to more distant encodings. Hence, success in sentence similarity tasks alone does not require explicit disentangled representation of semantic structure in the encoding. As research has so far focused on such tasks [55, 69, 255], the results do not shed much light on this matter.

The issue discussed here is related to a prior theoretical problem observed by Fodor and McLaughlin [92] concerning techniques for representing compositional structure via vector operations (their paper specifically addressing Smolensky’s suggestion [274]). They note that while it might be possible to achieve vectors that are deterministically mapped from semantic constituents and their mode(s) of combination, this is not sufficient if the vectors are opaque with respect to those component meanings. For semantic composition to be computationally tractable, components should not only be deterministically related to the vector representing the com-
plex, but present in it such that they are accessible to further operations. This is essentially the matter of (dis)entanglement reviewed above. Without taking a strong stance on Fodor and McLaughlin’s positions here, it suffices to note that similar issues continue to arise in state-of-the-art sentence encoding schemes, and require further scrutiny in future work.

A related problem with compositionality concerns the stability of token representations across different contexts. Transformers use contextual embeddings for input tokens, which means that the same token will receive different embeddings in different sentences. Consequently, even if the effects of each token embedding could be disentangled from the resulting sentence encoding, the embeddings themselves would depend on their surrounding environment. For example, Mickus et al. [196] demonstrated that word embeddings by BERT can vary even based on semantically irrelevant properties, such as whether the word occurs in the first or second sentence in a sentence pair. While context-dependent (sub)word representations can be highly useful for many NLP purposes [229], they also introduce further challenges for the explicit representation of compositional semantics.

Interim summary: the role of DNNs in semantic representation
Embeddings are highly useful for efficient and high-coverage word representation, and for finding words that are semantically related in some way. However, they are limited in at least two respects. First, embeddings do not reliably represent the nature of the semantic relation between embedding neighbours. If detailed semantic relations (like equivalence, hypernymy or antonymy) need to be distinguished, manually annotated lexical knowledge bases are still needed. Embedding neighbourhood metrics are necessary for obtaining sufficient coverage for semantic comparison tasks, but cannot provide a sufficient level of detail alone.

Second, representing compositional meanings is a challenge for embedding-based approaches. Straight-forward vector combination is not order-sensitive; and while asymmetry injection via e.g. weighing is possible [202], it cannot plausibly capture the range of semantic detail exhibited by symbolic formalisms for LFs. Effects of the surrounding context are recorded in more detail by encoder networks such as RNNs or Transformers. However, especially the increased use of attention has made it more challenging to identify any single encoding for the whole input sequence. Indirect methods for such pooling include the [CLS] token in BERT, and various vector combination techniques similar to those used

---

3Fodor and McLaughlin’s [92] discussion is mostly not directly related to compositionality but systematicity in cognition: e.g. the ability of thinking $R(a, b)$ implying the ability to think $R(b, a)$ [93]. Compositionality is one possible explanation for systematicity (and, according to them, the only feasible one). The issue is highly contentious and has elicited much discussion [8, 49, 194], which I do not engage with here; but it is likely that the extent to which DNNs can represent compositional structure has important consequences for it as well.
for embeddings. Such pooled sentence encodings have not shown good performance when applied directly to sentence similarity comparison [255]; but it is possible to adapt them specifically to such tasks, resulting in the encodings of semantically similar sentences becoming closer [55, 69, 255]. These supervised techniques rely on manually labeled parallel corpora as training data (e.g. SNLI).

In my judgement, the notion that explicit representation of compositional structure would arise in sentence encodings is not supported by either theoretical or empirical considerations. This does not mean that the encoder networks are doing anything “wrong”; rather, it means that they likely use different methods to achieve their goals than the construction of structured compositional representations in the latent space. If correct, these considerations indicate that DNN-based and symbolic semantic representation techniques are largely complementary, and hence neither is likely to subsume the other. This invites a (re)consideration of hybrid techniques that combine aspects of both in a non-reductive way [26, 98, 103, 170, 256]. One direction to take in such research is to optimize symbolic semantic representation formats to be directly combinable with (separately obtained) word embeddings. This is not a priority for any of the prominent formats reviewed in Section 4.1.1, which motivates a consideration of alternatives.

4.2 EAT

I now turn to the EAT format, which aims at maximally simple and transparent syntax-semantics mapping that is nevertheless expressive enough to provide a reasonable alternative to representations like DRT, AMR, or MRS. EAT is not intended to be a complete replacement of any prior representation format, but instead a useful addition to the range of possible techniques, with certain important novelties and benefits. Sections 4.2.1–4.2.3 cover the theoretical motivation of EAT from a linguistic perspective, with a focus on the syntax-semantics interface. Section 4.2.4 then presents the format itself.

4.2.1 Linguistic background

EAT is in large part based around linguistic ideas formulated within the generative framework [56, 57, 162, 212]. However, the format itself is agnostic about the linguistic formalism used, and has been implemented for both phrase structure-grammar and dependency grammar (Section 4.3.2). There is also no principled contradiction between EAT and alternative linguistic frameworks like cognitive grammar [160, 161] or construction grammar [74, 101, 286]. Investigating the relation between EAT and different linguistic theories is an interesting venue for future work.
Generative analyses treat sentences as *phrase structures* that can be formalized as trees. A simple example is shown in (13), where executing the phrase structures rules (13a-e) yields the sentence on the right. I adopt the standard notation of *XP* for a phrase the *head* of which is *X*. Roughly, the head determines the syntactic nature of the phrase. For example, the combination of the verb and noun *saw Mary* acts like a verb, not like a noun. Hence, it is headed by the verb, i.e. is a *verb phrase (VP)*. Similarly, *noun phrases (NPs)* are headed by nouns.

**Early variants of the theory** treated phrase structures as generated by language- and construction-specific rules [56, 57]. This also meant that there was no proper foundation for the syntax-semantics mapping beyond descriptive generalizations. Subsequent developments attempted to both simplify the underlying phrase structure rules, and provide a more systematic link between syntactic positions and semantic roles. The most prominent attempts at both were the *X*-theory [58, 132] in syntax and the *Uniformity of Theta Assignment Hypothesis (UTAH)* in the syntax-semantics mapping [18]. Both were central to the so-called *Government & Binding theory (G&B)* dominant in the 1980s and 1990s [59, 115].

*X*-theory maintains that each phrase (XP) is a projection of a head (X), and can contain three types of additional phrases: *complements, specifiers, and adjuncts*. Complements are directly attached to the head, which results in an intermediate or “bar”-level phrase (X’). Specifiers attach to the bar-level phrase to make up the full phrase. Adjuncts are optional modifiers attached to the bar-level that recursively project another bar-level phrase. I denote the specifier of a phrase XP as *Spec,XP* and the complement of XP as *Comp,XP*. The X’-schema is shown in (14).
It is worth noting that strict X'-theory has largely been abandoned in the post-G&G minimalist framework [32, 61, 62, 128, 248]. In minimalist analyses, X'-theory is no longer taken axiomatically, and is instead typically accounted for as an emergent consequence of more fundamental syntactic operations. Still, in practice it has remained a central heuristic device for syntactic analysis. One concrete consequence of its minimalist reinterpretation has been the elimination of non-branching X'-positions. I follow this in the notation for the remainder of this chapter, where X'-positions are only marked if a complement is present.

The shift from construction-specific phrase structure rules to the generic X'-theory severely limits the types of phrases allowed. While restricting the range of possible analyses on the surface-level, this is beneficial from a standpoint that seeks to link possible syntactic structures to interpretations in other parts of the linguistic system. One side of this analysis concerns linearization in mapping the syntactic representation to a phonological string [142], and the other concerns the mapping of the phrase structure to a semantic representation. I discuss the latter here.

Baker [18] suggested that thematic roles ("theta-roles" in generative lingo) correspond to the X'-schema in a systematic way. This is known as the Uniformity of Theta Assignment Hypothesis (UTAH). In clauses, a common formulation of UTAH is allocating Agent to Spec,VP and Theme to Comp,VP, as shown in (15) together with an example sentence.4

---

(14)

\[
\begin{array}{c}
\text{XP} \\
\text{YP} \\
\text{Specifier (= Spec,XP)} \\
\text{(X')} \\
\text{(Adjunct) (WP)} \\
\text{X'} \\
\text{X} \\
\text{ZP} \\
\text{Head Complement (= Comp,XP)}
\end{array}
\]

---

(15)

\[
\begin{array}{c}
\text{VP} \\
\text{Agent V'} \\
\text{V Theme} \\
\text{NP John} \\
\text{V NP saw Mary}
\end{array}
\]

---

4This variant of UTAH assumes the now-standard VP-internal subject hypothesis [155, 276], which differs from an earlier formulation where subjects originated above the VP [60].
Clauses are typically treated as extended projections of verbs [112, 113], built on top of a VP with additional functional heads [2, 60, 67, 68]. The VP is the complement of a functional head originally called I for inflection [59], and later T for tense [238]. It hosts verbal inflectional material like tense or mood (sometimes as overt auxiliaries), but allows cross-linguistic variation in specific interpretation [257, 295]. A higher functional head C then hosts complementizers (e.g. that) and assigns discourse-level features like force (declarative, question, command), topicalization, or focus [258].

Both TP and CP project X'-structure, and their specifier positions function as landing sites for movement of elements from inside the VP. In English, Spec,TP is the grammatical subject position hosting the Agent in active clauses (John saw Mary) and Theme in passive clauses (Mary was seen). Spec,CP hosts wh-phrases in questions (Who did John see?), and focused elements in the left periphery (Him John saw, not her). As an addition to phrasal movement, head-movement [87, 189] can e.g. displace an auxiliary from T to the higher C-position, as in many questions (Did John see Mary?). Example (16) shows the functional positions in the English clause (i.e. extended VP) with their prototypical contents.

(16)

```
CP
  Spec,CP (wh-phrase; focused phrase)
  C' (clause type; complementizer)
    C (subject)
      Spec,TP
    T' (tense; auxiliary)
      T (VP (...))
```

C and T have further been expanded to additional functional heads, culminating in the framework known as syntactic cartography [25, 66, 67, 68, 143, 258]). There is some tension between the cartographic augmentation of (16) and the minimalist paradigm that aims at simplifying syntactic operations and the resulting structures [32, 61, 62, 128]. One common strategy has been to treat C and T as notational shorthands for domains that open up to multiple functional heads [114, 250, 288, 295]. For example, the English T-domain includes positions for at least negation and modal auxiliaries in addition to tense. This idea is schematically shown in (17) (specifier positions omitted).
Table 4.2. Syntactic positions in the English clause.
(“T/C”: some functional head in the T/C-domain)

<table>
<thead>
<tr>
<th>Type</th>
<th>Role</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>original</td>
</tr>
<tr>
<td>head</td>
<td>verb</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>grammatical marker</td>
<td>auxiliary, negation</td>
</tr>
<tr>
<td></td>
<td>complementizer</td>
<td>C</td>
</tr>
<tr>
<td>phrase</td>
<td>grammatical subject</td>
<td>active voice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>passive voice</td>
</tr>
<tr>
<td></td>
<td>(direct) object</td>
<td></td>
</tr>
<tr>
<td></td>
<td>optional modifier</td>
<td>adjunct</td>
</tr>
<tr>
<td></td>
<td>(e.g. adverb or PP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>wh-phrase or focused</td>
<td>(any)</td>
</tr>
</tbody>
</table>

Without discussing the generalizability of (17) across languages here [250, 295], I adopt it as a useful analytic tool at least for English. It provides a classification procedure for allocating clausal elements to syntactic roles based on surface-level position and semantic interpretation, along with other grammatical features of the clause (e.g. voice). Table 4.2 shows the taxonomy. The clause is built from the V-head by adding thematic arguments (and possible adjuncts) inside the VP, moving the grammatical subject to Spec,TP, moving possible wh-phrases or focused phrases to Spec,CP, and applying head-movement from T to C in certain constructions.

Nouns also have a thematic argument-like position marked with the clitic ’s, which receives different interpretations depending on the noun [2, 10]. With event-denoting nouns it can receive an Agent or Theme reading, which is sometimes ambiguous (e.g. John’s eating). With object-denoting nouns it is usually interpreted as a possessor (e.g. John’s house), although
other possibilities are sometimes available (e.g. *John’s book* with the Agent reading ‘a book written by John’). I call this position simply *possessor*, and make no further commitments to its syntactic analysis beyond assuming it to reside somewhere in the (extended) noun phrase: either in the specifier of the noun itself [58, 132] or some functional projection [2, 4, 263].

Nouns can be modified by *adjectives* and verbs by *adverbs*, both of which can be analyzed as syntactic adjuncts. An alternative view within the cartographic approach is that at least some adjectives or adverbs occupy specifier positions of dedicated functional projections [66]. The differences between these variants are not directly relevant for present purposes, but they make different predictions about e.g. word order restrictions and scope relations in more detailed analysis. An adjective or adverb can also appear as the complement of a copula verb (*be*). Adjective/adverb phrases have a functional projection that determines *comparison class* (positive/comparative/superlative) [31], and can also host *degree adverbs* (e.g. *very*). Like with noun phrases, I remain agnostic about the more detailed analysis of these constructions.

Hale and Keyser [117] argued that *prepositional phrases* (PPs) function similarly to verb phrases in terms of argument structure. Like transitive verbs, prepositions relate two arguments, one of which is syntactically above the other. Standardly this has been analyzed as the prepositional phrase being an adjunct of the phrase it modifies; while Hale and Keyser argued both arguments to belong to the prepositional phrase according to X'-structure. Examples of both alternatives are given below in (18) for the same noun phrase; the left-hand variant showing the adjunction analysis and the right-hand variant showing the alternative where both arguments are inside the PP.

(18)

Both analyses result in a similar hierarchical structure, presented in a general form in (19). I denote the preposition’s arguments as *Figure* and *Ground*, based on prior linguistic literature [280, 282].
A similar three-fold analysis can be extended to connectives. These combine two elements into a complex element of the same syntactic type, as in (20), which connects two clauses into a larger clause.

(20)

\[
\text{John is glad} \quad \text{if Mary wins the race}
\]

The analysis has also been applied to coordination structures beyond clausal connectives [134, 142]. Marking both with CONN, the generic schema (21) holds, where X and Y are syntactic elements of the same kind (heads or phrases with the same type of head).

(21)

\[
X \quad \text{CONN} \quad Y
\]

A recurring pattern can thus be seen, where a head takes two arguments: one lower and the other higher in the structure. UTAH captures this in verb semantics directly via X'-theory, and similar analyses can be extended to prepositional phrases, connectives, and coordination structures. To abstract away from the strict X'-theory, a more generic scheme can be formulated with three roles in an asymmetric relation, shown in (22). I call the lower argument position of a head X its internal argument (Int,X), and the higher position its external argument (Ext,X). Note, however, that the arguments cannot be defined purely relationally as Ext,X being above Int,X, since Ext,X can also appear without Int,X in e.g. intransitive verbs.

\[\text{Int}/\text{Ext},X \text{ instead of Int}/\text{Ext},XP \text{ to avoid taking a stance on the label of the phrase containing the arguments. At least Ext,X might appear in a phrase not labeled by X; see the discussion on PPs.}\]
Crucially, (22) can also manifest in constructions where the external and internal argument do not map to the complement and specifier in X'-theoretic terms. For example, even if PPs are treated as adjuncts (see above), the element they modify is still the external argument of the P-head despite not being its specifier. Also, in more recent theories of verbal argument structure, thematic positions are often not assigned via X'-theoretic UTAH but instead by dedicated syntactic positions within an extended VP projection [3, 5, 34, 35, 177, 249]. Still, verbal arguments continue to systematically map onto thematic arguments based on their syntactic positions where the internal argument resides below the external argument, satisfying (22). The purpose of (22) is thus to abstract away from such syntactic details without losing the explanatory benefits of UTAH-type restricted syntax-semantics interface links.

In addition to the clausal arguments specified in Table 4.2, we now have a restricted set of possible syntactic roles for a broader range of linguistic elements. We can ask of each element whether it is (i) an argument-taking head (verb, preposition, connective), (ii) an internal argument of a head, (iii) an external argument of a head, (iv) an adjunct, or (v) a functional head in an extended projection. The next stage is assigning each type of element a dedicated rule for semantic interpretation.

### 4.2.2 Conjunctivism

I cover the basics of the conjunctivist framework developed by Pietroski [129, 231, 232, 233, 234, 235]. In certain parts my analysis departs from Pietroski’s, and I discuss this when needed. I still maintain the term conjunctivism, which I take to denote a broad framework that allows multiple variants.

To satisfy compositionality, the syntax-semantics interface must function in a manner that can determine the meaning of a complex expression based on its syntactic constituents and their mode of combination. In Section 4.2.1 I went through the relevant background concerning the syntactic

---

6The qualification of “can determine” (as opposed “necessarily determines”) is important here, as it retains the possibility of non-compositional semantics for e.g. idioms. Emphasizing compositionality in natural language does not require denying other, non-compositional aspects of meaning. Rather, the point is to make note of the combinatorial nature of many (in fact infinitely many) expressions, not only in grammar but semantic interpretation as well.
aspects of both. Syntactic heads can be *lexical* or *functional*, and a *phrase* is an *extended projection* of a lexical head with additional functional heads [112, 113]. Syntactic combination comes in three main kinds: (i) adding arguments according to the scheme (22); (ii) adding functional heads to an extended projection of a lexical head; and (iii) adjunction. The *minimalist* framework in syntactic theory [32, 61, 62, 128, 248] has attempted to further unite different variants of syntactic combination, but this three-fold distinction is useful at least on a descriptive level for present purposes. Another set of operations concerns *movement* that can have e.g. scopal effects on semantic interpretation, but I abstract away from this here and interpret elements in their pre-movement positions that determine argument structure (Section 4.2.1; Table 4.2).

Conjunctivism is based around two related main claims, both aiming at significantly simplifying the syntax-semantics link. First, possible semantic types are severely restricted; and second, semantic combination is primarily accounted for by conjunction instead of function application. However, with more complex aspects of semantics some limited forms of function application need to be (re)introduced. I start with the basic variant first, and subsequently discuss additions and modifications for more complex cases.

*Lexical heads* (verbs, nouns, adjectives, adverbs) are interpreted as *monadic predicates*. Common nouns and (subsective) adjectives are straight-forwardly analyzed as monadic predicates over e.g. objects or masses. It is also possible to change any logical constant $c$ to a predicate $P(x)$ via the equivalence postulate $P(x) \iff x=c$ [232, 245, 246], which allows treating proper names as predicates without introducing the problems in description-based accounts of names [157]. Neo-Davidsonian semantics further treats verbs as predicating a single event variable irrespective of transitivity, and allows extending this analysis for (subsective) adverbs and event-denoting nouns. The first conjunctivist tenet is thus that the semantic type of a lexical head is a *monadic predicate*, i.e. $<e,t>$.

*Phrases* are built from lexical heads with thematic arguments, additional functional projections, and optional adjuncts. The second major claim of conjunctivism is that phrases are also monadic predicates. The consequence of this is that semantic combination must *map monadic predicates to others*. One of the simplest ways to do this is *conjunction*, where the predicates $P(x)$ and $Q(x)$ make $P(x) \land Q(x)$. As its name suggests, conjunctivism takes conjunction to be the most central semantic mode of combination. In particular, it serves as the *default* interpretation of the combination of two syntactic objects. This applies in at least two kinds of cases: *adjunction*, and the adding of *functional heads* to an extended lexical projection. When an adjunct or functional head is a monadic predicate (see below for discussion of exceptions), it is simply conjoined with the predicate linked to the lexical head(phrase) it modifies, as in (23).
The simplest conjunctive rule (23) is the backbone of conjunctivism as the default rule applied when no more specific rule is available. Such more specific rules determine argument relations needed to account for thematic roles of verbs as well as relations expressed by prepositions or connectives. The third main claim of conjunctivism is that such rules are limited to determining dyadic relations assigned to dedicated syntactic constructions. To maintain the second tenet (that complex phrase are monadic predicates), a further operation of existential closure is used to limit the range of free variables involved.

With verbs, Ext, V corresponds to the Agent and Int, V to the Theme, and both arguments are existentially bound. With prepositions, Ext, P and Int, P correspond to the two relata connected by the preposition, and the latter is existentially bound. This results in the prepositional phrase being a complex predicate of Ext, P: e.g. *a dog in a house* being a predicate satisfied by certain types of dogs. The VP case if shown in (24a) and the PP case in (24b), with the schematic syntactic structure on the left and the corresponding semantic interpretation on the right. (“AG” and “TH” are abbreviations of “Agent” and “Theme”, respectively.)

If neither (24a) nor (24b) applies, the default rule (23) results in a simple conjunctive interpretation. Using the term SEM for semantic interpretation [62], I summarize the main tenets of basic conjunctivism in (25):

(25)  

a. SEMs of lexical heads (nouns, verbs, adjectives, adverbs) are monadic predicates.

b. SEMs of complex phrases are monadic predicates.
c. Arguments of verbs/prepositions are assigned by dedicated syntactic positions as shown in (24).

d. When two syntactic objects are combined and one is not in an argument position of the other, the SEM of the combination is the conjunction of their SEMs applied over the same free variable, as shown in (23).

The tenets (25a-d) are implemented in the EAT format (Section 4.2.4). However, when analysis is broadened to linguistic constructions beyond simple examples, it quickly becomes evident that they alone are unable to account for all semantic phenomena. In Section 4.2.3 I review a range of such cases, and lay out possible techniques for extending conjunctivism to cover them while remaining faithful to the theoretical goal of minimizing semantic types and modes of combination.

4.2.3 Refinements to conjunctivism

Discussion of non-conjunctive aspects of semantics quickly becomes highly complex, and – as Pietroski himself has repeatedly pointed out [232, 235] – conjunctivism is a work in progress. In this section I briefly review possible directions the analysis could take to account for such extensions. Some are based on Pietroski’s own work, and in these cases I refer to him explicitly. When I do not, the suggestions are my own (although not particularly original, building on other work cited). In Section 4.2.4 I discuss how these considerations relate specifically to the EAT-format.

Unaccusativity
Intransitive verbs can be divided into unergative and unaccusative variants, depending on the semantic interpretation of their grammatical subject [47, 166, 228]. In unergatives the subject is interpreted as the Agent (e.g. *run*), and in unaccusatives as the Theme (e.g. *break*). In syntactic accounts this is often taken to arise from two underlying syntactic positions, which then surface as the same “subject” position after syntactic movement [35, 117, 232, 249]. This fits nicely with the syntactic picture explained in Section 4.2.1, where the original argument position (Ext,V or Int,V) is different from the (English) grammatical subject position: Spec,TP. An alternative would be to assign unaccusativity/unergativity as a lexical feature specific to each intransitive verb [166].

Plurality
Following classical predicate logic, so far I have assumed that variables over which predicates are applied take single entities (e.g. an object or event) as possible values. This provides no satisfactory way to analyze the semantics of number (singular/plural), numerals, and verbs with collective readings only satisfied by a group of entities together as the Agent.
Pietroski [231, 232, 235] uses *plural variables* [33, 265] to extend the conjunctivist analysis. Unlike standard singular variables, these can take multiple values. I distinguish them from singular variables via capitalization, and denote the membership of a singular entity in a plurality by “≺”. A plural variable may also have only one value.

For each singular predicate $P$, we can introduce a corresponding plural predicate $P^\ast$ such that $P^\ast(X) \iff [x \prec X \rightarrow P(x)]$. There are also *collective predicates* that are only satisfied by pluralities. Numerals are examples: e.g. *two dogs* can be analyzed as $\text{DOG}(X) \land 2(X)$, where $\text{DOG}(X)$ means that each $x \prec X$ is a dog, and $2(X)$ means that $X$ has two values. Similarly, grammatical *number* can be treated as restricting the plural variable to having only one value (singular) or having multiple values (plural). Some verbs (e.g. *gather, rain*) require a collective reading for the plural Agent. The analysis can further be extended to event pluralities and subevents [177, 265], but I will not go further into this here. For present purposes is suffices to follow Pietroski in treating variables as taking plural instead of singular values. I assume this (implicitly) in the remainder of this chapter.

**Predicate modifiers**

As discussed in Publication V, many non-conjunctive elements can be analyzed as *predicate modifiers* of the semantic type $<<e,t>,<e,t>>$. Including these also requires adding function application as a mode of combination, taking the scheme closer to the Montagovian approach – albeit still remaining much more limited.

**Negation.** The Montagovian/Fregean analysis treats negation as switching a truth-value from *True* to *False* or vice versa, and hence having the semantic type $<t,t>$ [122]. However, the present conjunctivist system does not introduce existential closure over the event variable in the clause that would give it the type $<t>$. Instead, the clause is a monadic predicate of the type $<e,t>$ with the free event variable, as shown in (24a). Furthermore, to keep in line with (25b), the result of applying negation should be a monadic predicate as well, since it is a complex phrase. Pietroski [235] provides an alternative analysis of negation, following Tarski [283]. He defines two “arrow” operators as follows: $\uparrow P(x)$ applies to everything if $P(x)$ applies to something, and otherwise applies to nothing; and $\downarrow P(x)$ applies to everything if $P(x)$ applies to nothing, and otherwise applies to nothing. The result of applying either arrow operator is a monadic predicate in semantic type, having one free variable. Pietroski [235] assigns them operators to the T-domain in English (Section 4.2.1). This fits well with the English negation apparently residing in the T-domain since it appears below auxiliaries.

**Modality.** In mainstream formal semantics, the main tool for dealing with modal statements involves *possible worlds* [122, 156]. Originally coined by Lewis [168], the main idea is that each statement is relativized to each
possible world, the nature of which can be left unspecified for present purposes.\footnote{Lewis' \cite{168} own controversial position was that possible worlds are all equally real in some concrete sense. A common alternative to this is to think of them as counter-factual scenarios of some kind \cite{157}. Both options are at least \textit{prima facie} problematic: the notion of “concrete” possible worlds is obscure to say the least; and “counter-factual” seems to be at least equally complex a notion as modality itself (or even to rely on it). One benefit of the analysis I present here (and in Publication V) is that it allows keeping more distance from questions of modal metaphysics than standard approaches. Since all I suggest is that modal operators are predicate modifiers in semantic type, I can retain agnosticism about how they should be analyzed at further stages of interpretation.} Hence, the type <t> is reanalyzed as <w,t>: a function from a possible world w to a truth-value. Now, modal operators can be interpreted as quantifying over the world parameter, resulting in a truth-value. Hence, they are of the type <<<w,t>,t>>. The two main operators indicate possibility and necessity, which are analyzed as existential and universal quantification over possible worlds, respectively.

Without denying that reference to possible worlds (or something like them) will be needed at some stage of modal interpretation (but see Footnote 7), it would be beneficial to avoid introducing them to the semantic types themselves. One way to achieve this is to treat them as predicate modifiers. This approach has at least three major benefits: (i) assimilating negation and modal operators in semantic type; (ii) assimilating syntactically varying modal operators in semantic type; and (iii) avoiding the introduction of possible worlds on this level of analysis.

**Non-subsective modifiers.** The predicate modifier analysis can be extended to other modifiers that to not entail the element they modify. These are called non-subsective, as the extension of the resulting interpretation is not a subset of the extension of element modified. For instance, an alleged criminal need not be a criminal, making alleged a non-subsective adjective. Another class of elements to be treated in this way are degree adverbials, which modify an adjective or adverb (e.g. very, somewhat).

**Connectives and coordination**

Some clausal connectives entail the clauses they connect (e.g. and, while), but others do not (e.g. if, or). At least for the latter, some non-conjunctive mode of combination is needed. The standard formal semantic account of connectives follows the classical truth-table definition \cite{296}, where the connective yields a truth-value from two truth-value arguments: <t,<t,t>>. For example, an inclusive disjunction (∨) yields True if at least one of its arguments is True, and False otherwise. The most straightforward modification is to change this to <<<e,t>,<<e,t>,<e,t>>, where the arguments are clausal predicates produced via the arrow operators (see above). Coordination structures are similar to connectives, but they connect non-clausal elements, such as individual lexical items (nouns, verbs, adjectives/adverbs) or phrases. In English (and many other languages),
a significant number of elements function as both connectives and coordinators, which invites a uniform treatment in semantics. Fortunately, this now comes essentially for free, as the standard treatment of coordination is that they map two predicates to a third, i.e. are of the type $<< e, t >>, << e, t >>, < e, t >>$ [122].

**Quantification**

Quantifiers (Q) take two arguments, called the *restrictor* and the *nuclear scope*. For example, in the sentence *All dogs ran*, the restrictor is *dogs* and the nuclear scope is *ran*. In syntax, a common generative analysis of quantification involves the assumption that all quantified phrases (QPs) appear above the verb via movement [190, 191]. The movement is typically phonologically *covert*, i.e. the lower copy is pronounced. This analysis allows the basic phrase structure scheme (22) to be extended to quantifiers as shown in (26), with the restrictor as the internal argument and the nuclear scope as the external argument.

(26)

```
NP  nuclear scope
     QP
     Q  restrictor
```

Pietroski bases his analysis of quantification on this affinity between QPs, VPs, and PPs [231, 232, 235]. His approach has certain theoretical benefits. Notably, it predicts the *conservativity* of determiners, which is a universal semantic property that lacks a principled explanation in the standard account [231, 232, 235]. On the other hand, it involves significant complexities that are not present in the conjunctivist analyses of VPs or PPs; the most evident being the introduction of a *sui generis* semantic type for quantifiers. Since Pietroski’s analysis is not implemented in the EAT-format (Section 4.2), I do not discuss it further here. Another possible alternative would be to assimilate quantifiers to connectives or coordinators in semantic type: $<< e, t >>, << e, t >>, < e, t >>$. A simpler analysis of some quantifiers is also made available by adopting plural predication (see above). For example, numerals are generally included in the range of quantifiers, but can be analyzed as plural predicates.

---

8Informally, conservativity means the following property of a determiner/quantifier Q: if $Q$ Ps are $W$, then $Q$ Ps are $P \wedge W$. For example, if *all dogs are animals*, then *all dogs are dogs and animals*; or if *some dogs are brown*, then *some dogs are dogs and brown*, etc. Non-conservative determiners have been suggested to be absent from attested natural languages [24], and children fail to learn artificial non-conservative determiners in experimental contexts while succeeding at otherwise comparable conservative ones [130]. A principled account of their impossibility in would thus be a significant theoretical virtue.
Mood / force
If clauses have the type \(<e,t>\) instead of \(<t>\), the question arises of when their free variable position gets bound; given that clausal meanings are evidently truth-evaluable and hence seem to be of the type \(<t>\) at least at some point of their semantic interpretation. Without going into detail on this question, I adopt the rough idea presented by Lohndal and Pietroski [178] that the C-domain (Section 4.2.1) makes the clausal meaning into an instruction for possible further uses, such as performing speech acts [12, 268]. Roughly, the declarative mood turns the clausal meaning into an instruction to make an assertion, the interrogative mood into an instruction to perform an inquiry, and the imperative mood into an instruction to perform a command. I assume that the remaining free variable is bound at the C-domain when grammatical mood/force is specified.9

Clausal arguments
Propositions are commonly thought of as truth-evaluable semantic contents, roughly corresponding to the meanings of declarative clauses [149, 150, 195, 275]. Some verbs, notably those depicting so-called propositional attitudes (believe, hope, doubt, etc.), take clausal arguments interpreted as propositions instead of events. Pietroski [230, 232] argues that some (propositional attitude) verbs take a Content argument instead of Theme. This obviously does not address the matter sufficiently, but still provides an appropriate starting point for future analysis.

4.2.4 Structure of EAT

This section describes the EAT-format, which aims at being an optimal NLP-implementation of the conjunctivist framework discussed in Sections 4.2.2–4.2.3.

Argument structure
The idea behind EAT is to assign words to three possible roles: E-, A-, or T. These terms come from the verbal Event, Agent, and Theme; but have different interpretations based on the construction. Words that do not directly have any of these roles are modifiers of elements that do, and are assigned to the same role as the element they modify. Syntactically, such modifiers are usually functional heads or adjuncts. The semantics of the three relevant constructions are given in (23)–(24); repeated in (27), where X is any syntactic object and “Modifier” is a functional head or adjunct. Rule (27a) applies for VPs, (27b) applies for PPs (and connectives/coordinators), and (27c) in the default case.

9Note also that since \(\uparrow P(x)\) or \(\downarrow P(x)\) is always true of either everything or nothing (see discussion on negation above), their existential and universal quantification are equivalent: i.e. \(\exists x \uparrow P(x) \iff \forall x \uparrow P(x)\); and \(\exists x \downarrow P(x) \iff \forall x \downarrow P(x)\). This means that all variable binding can be existential, which further simplifies the theory.
The aim of the EAT format is to build this three-fold scheme into the semantic representation in a way that maximally retains the sentence’s meaning. Because three is such a small number (in comparison to e.g. the \( \sim 100 \) semantic roles in AMR [19]), it is possible to use positional encoding for the roles. The main unit of analysis in EAT is an EAT-triplet, which contains the E-, A-, and T-roles concatenated in this order. If a role is not present, it is marked with the null placeholder \( \emptyset \) to maintain consistent size for the triplets. Some examples are shown in (28).

(28) a. A dog sees a cat
   \(<\text{see, dog, cat}>\>

b. A dog walks
   \(<\text{walk, dog, } \emptyset >\>

c. A cat is seen
   \(<\text{see, } \emptyset , \text{ cat } >\>

If an element in an EAT-sequence is modified by another word, this modifier is appended to another EAT-triplet to the corresponding role position. The new EAT-triplet is then concatenated to the list of EAT-triplets after the original triplet. The new triplet is again forced to the fixed size by adding empty tokens to positions that lack material. However, if multiple parts of the original triplet are modified, these are included to the same modifier triplet, as in (29).

(29) A brown dog sees a black cat
   \(<\text{see, dog, cat}> <\emptyset, \text{ brown, black}>\>
Prepositions occupy the E-role like verbs, where the A-role is Ext,P (Figure) and the T-role is Int,P (Ground). To increase generality, an alternative for repeating the same word twice is to mark only the prior role of the A-role. In (30), the PP in a house modifies dog, which is the A-role of see; and the PP on a roof modifies cat, which is the T-role of see. Hence, the first P modifies a prior A-role, and the second PP modifies a prior T-role. The current EAT implementation uses the more general variant, where the prior role is marked on the preposition as a grammatical feature (Table 4.3) and the A-role is left empty, shown in the last line of (30).

(30) A dog in a house sees a cat on a roof
   \(<\text{see}, \text{dog}, \text{cat}> <\text{in}, \text{dog}, \text{house}> <\text{on}, \text{cat}, \text{roof}>
   <\text{see}, \text{dog}, \text{cat}> <\text{in}_A, \emptyset, \text{house}> <\text{on}_T, \emptyset, \text{roof}>

Indirect objects typically correspond to a Recipient argument, which is often complementary with a PP headed by to.\(^{10}\) To facilitate the semantic unification of syntactic paraphrases, I assimilate both to the to-PP in EAT, shown in (31). This allows us to retain the three-fold EAT-structure instead of adding Recipient as another thematic role on par with Agent and Theme.

(31) Mary gives John a flower / Mary gives a flower to John
   \(<\text{give}, \text{Mary}, \text{flower}> <\text{to}_E, \emptyset, \text{John}>

Connectives have the same structure as prepositions (Section 4.2.1). With clausal connectives as in (32), the arguments are represented by verbs, which themselves project their own EAT-triplets. The same strategy is applied with clausal arguments, where the verb represents the argument and later manifests its own arguments in a separate EAT-triplet.

(32) a. John is glad if Mary wins
   \(<\text{be}, \text{John}, \text{glad}> <\text{if}_E, \emptyset, \text{win} > <\text{win}, \text{Mary}, \emptyset>
   b. John believes that Mary wins
   \(<\text{believe}, \text{John}, \text{win} > <\text{win}, \text{Mary}, \emptyset>

\(^{10}\)There are certain constructions where the complementarity fails to hold, such as (1). (I denote ungrammaticality with an asterisk.) Such exceptions indicate that the grammatical relation between datives and to-constructions is more complex than simple equivalence, and hence suggest against syntactic analyses that derive one from the other.

(1) a. John makes Mary a pizza
   b. *John makes a pizza to Mary

I do not consider this observation to be sufficient to oppose their assimilation in EAT; but it would of course be trivial to differentiate the dative-variant and standard variant of to as separate words, since their distinction is available from surface syntax. In languages with more variants of such applicative constructions where PPs are promoted to a grammatical object position [242], more “pseudo-PPs” like this would be needed. Like dative Recipients in English, such more elaborate applicatives would thus also be compatible with EAT’s three-fold structure.
Table 4.3. Grammatical features in EAT.

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Features</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbal/clausal</td>
<td>Force</td>
<td>declarative, imperative, question</td>
</tr>
<tr>
<td></td>
<td>Negation</td>
<td>affirmed, negated</td>
</tr>
<tr>
<td></td>
<td>Voice</td>
<td>active, passive</td>
</tr>
<tr>
<td></td>
<td>Tense</td>
<td>present, past, perfect, pluperfect</td>
</tr>
<tr>
<td></td>
<td>Aspect</td>
<td>perfective, imperfective</td>
</tr>
<tr>
<td>Prepositional</td>
<td>Prior role</td>
<td>E, A, T</td>
</tr>
<tr>
<td>Nominal</td>
<td>Number</td>
<td>singular, plural</td>
</tr>
<tr>
<td></td>
<td>Definiteness</td>
<td>definite, indefinite</td>
</tr>
<tr>
<td></td>
<td>Possessive</td>
<td>possessive, non-possessive</td>
</tr>
<tr>
<td>Adjectival/adverbial</td>
<td>Degree</td>
<td>positive, comparative, superlative</td>
</tr>
</tbody>
</table>

Grammatical features
Grammatical features are additionally assigned to each word in the EAT-triplet, listed in Table 4.3. Verbal/clausal and prepositional features apply to the E-role; nominal features are divided between the A-, and T-roles; and adjectival/adverbial features between all three roles. Like the argument structure for constructing the EAT-triplets, we obtained the features from the syntactic parse of the original sentence. Both probabilistic context-free grammar parses (with POS tagging) and dependency grammar parses contain all information necessary for their collection. EAT has been implemented for both syntactic formats, as presented in Publication V.

Possible refinements
EAT as defined so far only implements the basic conjunctivist theory discussed in Section 4.2.2. However, in Section 4.2.3 I mentioned several complications that called for additions to the system. I will now quickly go through how these relate to EAT in its present form and possible future developments.
Plural variables. The EAT-format itself does not contain variables, but EAT-triplets can be analyzed as containing free variables for E-, A-, and T-roles as specified in (27). Since this is not part of EAT as such but rather concerns its interpretation by a human reader, it is possible to treat these variables as plural without affecting the format.

Unaccusativity. Unaccusative verbs could be readily accounted for as verbs lacking an Agent in the active voice. The main problem with unaccusativity is not EAT's inability for representing it, but rather its unavailability in surface-level syntactic parses of the kind used in the experiments discussed in Section 4.3. Hence, in the parse-to-EAT derivations currently implemented (see Publication V for details), the subject of all intransitive verbs is assigned to the A-role. However, the EAT-format itself would allow assigning it to the T-role in unaccusatives if this information was available from the parse or some external lexical resource.

Predicate modifiers and connectives/coordinators. As discussed in Section 4.2.3, many complications to the basic conjunctivist scheme can be accounted for by introducing second-order functions that take monadic predicates as arguments and yield another monadic predicate: \(<e,t>,<e,t>>\) and \(<e,t>,<<e,t>,<e,t>>\). Since these two variants share the monadic-dyadic distinction with the more basic types, adding a single Boolean feature for first- vs. second-order interpretation would suffice to mark this four-way taxonomy in EAT. Subsective modifiers (e.g. nice, happy) and non-subsective predicate modifiers (e.g. possible, alleged) are both monadic; while basic dyadic relations (e.g. in, on) and connectives/coordinators (e.g. if, or) are both dyadic. Hence, adding an additional Boolean marker for the first- vs. second-order interpretation would allow extending EAT to all four types. However, like unaccusativity, this information is unavailable from the syntactic parse alone, which is why it is not included in the current implementations used in the experiments.

Quantification. The present version of EAT treats quantifiers as normal modifiers. This fit well with those quantifiers that allow analysis as plural predicates, but more elaboration would be required to differentiate these from quantifiers that resist this approach (e.g. every, no). This deficiency does not have practical repercussions for the experimental tasks reviewed in Section 4.3. Furthermore, such quantifiers form a small closed class in English. Hence, unlike with e.g. non-subsective modifiers (an open class), recovering their presence is feasible simply via keyword lists.

Clausal arguments. The analysis of non-entailed propositional arguments was left as an open question in Section 4.2.3, with the tentative view that some verbs take Content rather than Theme arguments [231, 232]. Again, this difference could be marked via a Boolean marker similar to the grammatical features. Like with unaccusativity and first- vs. second-order interpretation (see above), its absence in the current EAT-implementation stems from its lack of marking in the syntactic parse.
Vectorization
The EAT-representation of a sentence is a sequence of EAT-triplets, which are of a fixed size (since the lack of an argument is marked with an empty token). Words are lemmatized, and their grammatical properties are allocated to Boolean features (0/1) derived from the syntactic parse. In addition to the features in Table 4.3, relative pronouns are also marked as referring to the prior role of their correlate. With prepositions, the A-role is left empty, and the prior role of the A-role is assigned to a grammatical feature. There are 28 Boolean grammatical features in the current implementation. More could be added based on the demands of the task.

After lemmatization, words are assigned to pre-trained embeddings. The current implementation uses 300-dimensional GloVe embeddings [225] derived from a Common Crawl corpus.11 These embeddings are provided with the large English model of the NLP library Spacy [127].12 The EAT-format itself is independent of the embedding matrix used. The EAT-triplet is then vectorized by concatenating the 28 Boolean grammatical features and the three 300-dimensional word embeddings. The vectorization of the entire EAT-representation is thus a sequence of 928-dimensional vectors.

4.3 Experiments
I summarize the experiments presented in more detail in Publication V. These concern the comparison of MRS to an EAT-inspired reduced variant (Section 4.3.1), retrieving parallel corpora based on detailed grammatical and semantic information (Section 4.3.2), and generating English from EAT (Section 4.3.3).

4.3.1 MRS vs. MRS-EAT for text reconstruction
Dependency MRS (DMRS) is a graph-representation of MRS (see Section 4.1.1), where nodes indicate word meanings and edges their semantic relations [70]. Unlike the original MRS, it eschews variables, which makes it easier to apply in certain NLP settings. Hajdik et al. [116] trained an LSTM-based NMT network to reconstruct original English sentences from DMRS linearized in the Penman format [107]. Publication V demonstrates that text reconstruction performance improves when semantic roles in DMRS are drastically reduced in ways motivated by the theoretical considerations reviewed in Sections 4.2.1–4.2.3. The original (D)MRS uses 52 roles, and there are two variants of the reduced version: one containing nine roles and the other three. I call these MRS_{52}, MRS-EAT_{9}, and MRS-EAT_{3}, respectively. Table 4.4 shows the conversion.

11http://commoncrawl.org/.
12https://spacy.io/
Table 4.4. Conversion between MRS52, MRS-EAT9, and MRS-EAT3 (from Publication V).

<table>
<thead>
<tr>
<th>MRS52</th>
<th>MRS-EAT9</th>
<th>MRS-EAT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1-NEQ</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>ARG1-H</td>
<td>A-CLAUSE</td>
<td></td>
</tr>
<tr>
<td>L-INDEX-NEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-HNDL-HEQ</td>
<td>L-CONN</td>
<td></td>
</tr>
<tr>
<td>L-HNDL-H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARG2-NEQ (of verbs)</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>ARG2-NEQ (of prepositions)</td>
<td>T-PREP</td>
<td></td>
</tr>
<tr>
<td>ARG2-H</td>
<td>T-CLAUSE</td>
<td></td>
</tr>
<tr>
<td>R-INDEX-NEQ</td>
<td>R-CONN</td>
<td></td>
</tr>
<tr>
<td>R-HNDL-HEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R-HNDL-H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prepositions (POS-tag “p”)</td>
<td>P</td>
<td>M</td>
</tr>
<tr>
<td>rest</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the linearized (D)MRS, lemmatized words are followed by their arguments in parentheses, first with the semantic role and then the word occupying that role. Modifiers can be complex and recursively contain their own arguments. An example MRS52 representation is shown in (33).

(33) The woman showed herself an office

\[
\begin{align*}
\text{ARG1-NEQ} & \text{ARG1-H} \\
\text{L-INDEX-NEQ} & \text{L-HNDL-HEQ} \\
\text{ARG2-NEQ (of verbs)} & \text{ARG2-NEQ (of prepositions)} \\
\text{ARG2-H} & \text{R-INDEX-NEQ} \\
\text{R-HNDL-HEQ} & \text{R-HNDL-H} \\
\text{prepositions (POS-tag “p”)} & \text{rest} \\
\end{align*}
\]

In MRS-EAT3, only the A- and T-roles are used, along with the remaining default class of modifier (M). The E-role is implicit; anything that takes an A- or T-role is by definition in the E-position with respect to them. For example, prepositional phrases are modifiers (M) that have their own A- and T-roles. MRS-EAT9 includes more detailed distinctions between the relations, by separating between verbs, prepositions, and connectives in A- and T-roles; and between prepositional and non-prepositional modifiers. Additionally, it includes separate markers for clausal A- and T-roles. In both MRS-EAT variants, the indirect object (ARG3-NEQ in MRS52) is changed to the to-preposition, and treated as a normal prepositional modifier. The MRS-EAT9 representation of (33) is shown in (34).

(34) The woman showed herself an office

\[
\begin{align*}
\text{ARG1-NEQ} & \text{ARG1-H} \\
\text{L-INDEX-NEQ} & \text{L-HNDL-HEQ} \\
\text{ARG2-NEQ (of verbs)} & \text{ARG2-NEQ (of prepositions)} \\
\text{ARG2-H} & \text{R-INDEX-NEQ} \\
\text{R-HNDL-HEQ} & \text{R-HNDL-H} \\
\text{prepositions (POS-tag “p”)} & \text{rest} \\
\end{align*}
\]
Table 4.5. Comparison between MRS$_{52}$, MRS-EAT$_9$, and MRS-EAT$_3$ in text reconstruction performance (from Publication V).

<table>
<thead>
<tr>
<th>Technique</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>Exact match</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRS$_{52}$</td>
<td>83.28</td>
<td>75.82</td>
<td>69.66</td>
<td>64.22</td>
<td>47.60</td>
<td>20.2%</td>
</tr>
<tr>
<td>MRS-EAT$_9$</td>
<td>83.66</td>
<td>76.23</td>
<td>70.13</td>
<td>64.76</td>
<td>48.05</td>
<td>23.8%</td>
</tr>
<tr>
<td>MRS-EAT$_3$</td>
<td>81.40</td>
<td>73.30</td>
<td>66.75</td>
<td>61.06</td>
<td>45.96</td>
<td>17.9%</td>
</tr>
</tbody>
</table>

Performance was measured with the BLEU score [220] between 1 – 4-grams, the METEOR score [20], and the exact match rate between original and reconstructed texts. Table 4.5 shows the results. Differences between the formats were small, despite the large divergence in the number of semantic roles they use. In BLEU and METEOR, the difference between the highest and lowest scores never exceeded four points. This indicates that even with a drastic reduction of semantic roles from 52 to only three, most of the relevant information is retained even though the alternative representation is significantly simpler. More surprisingly, MRS-EAT$_9$ improved performance from MRS$_{52}$ in all metrics. A likely reason for this is the increased generalizability of the model through the assimilation of rare relation types to more common ones.

The results of text reconstruction with MRS-EAT illustrate that semantic representation in NLP has room for far more reduction in the number of semantic roles compared to prior representation formats. Expressiveness should not come at the expense of simplicity, given the relevance of the latter for implementation and application. Strategically, it seems sensible to start with the absolute minimum of roles, and work upward when increasing expressiveness truly calls for it. Uniquely among semantic representation formats in NLP, EAT is built around this premise.

4.3.2 Parallel corpus extraction

Apart from the MRS-EAT experiments (Section 4.3.1), all other empirical evaluations of EAT in Publication V use the EAT-format discussed in Section 4.2. This section discusses information retrieval from a syntactically parsed corpus; specifically parallel corpus extraction between grammatical classes that share the same argument structure but differ in grammatical features (see Section 4.2.4; Table 4.3).

The datasets used for these experiment were the Stanford Natural Language Inference corpus (SNLI) [37], and English sentences from the Stanford Neural Machine Translation (SNMT) corpora [182, 183]. SNLI is pre-parsed with the Stanford probabilistic context-free grammar (PCFG)

---

https://nlp.stanford.edu/projects/snli/  
https://nlp.stanford.edu/projects/nmt/
Table 4.6. Parallel corpora between grammatical classes, derived from the SNLI and SNMT corpora (from Publication V).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value pairs</th>
<th>No. pairs</th>
<th>Example pair (from SNLI)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SNLI</td>
<td>SNMT</td>
</tr>
<tr>
<td>Aspect</td>
<td>simple progressive</td>
<td>5753</td>
<td>4436 A black dog digs in the snow. A black dog is digging in the snow.</td>
</tr>
<tr>
<td>Tense</td>
<td>present</td>
<td>1061</td>
<td>19674 The dog runs through the water. the dog ran through the water</td>
</tr>
<tr>
<td></td>
<td>past</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negation</td>
<td>affirmed</td>
<td>858</td>
<td>15317 The man has a musical instrument. The man does not have a musical instrument.</td>
</tr>
<tr>
<td></td>
<td>negated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voice</td>
<td>active</td>
<td>57</td>
<td>61 A woman is hitting a tennis ball. A tennis ball is being hit by a woman.</td>
</tr>
<tr>
<td></td>
<td>passive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tense</td>
<td>present/past</td>
<td>27</td>
<td>3711 A man falls down A man has fallen down.</td>
</tr>
<tr>
<td></td>
<td>(plu)perfect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>declarative</td>
<td>1</td>
<td>9648 You are coming with me for coffee. Are you coming with me for coffee.</td>
</tr>
<tr>
<td></td>
<td>question</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parser [153], and dependency parses were created from the SNMT corpus with Spacy. Publication V provides the rules for mapping both PCFG and dependency graphs to EAT. From a theoretical perspective, these are based on first assimilating the parses to the kinds of syntactic structures discussed in Section 4.2.1.

The purpose of parallel corpus extraction was to combine two central aspects of linguistic information retrieval which EAT optimizes: detailed similarity comparison, and separating grammar from argument structure. The parallel corpora contain sentences that share argument structure and all except one grammatical feature. Table 4.6 shows the feature pairs and the sizes of the parallel corpora, which we provide as open-access.

4.3.3 Text reconstruction and transformation

EAT2seq is an encoder-decoder network that maps a vectorized EAT to English, and has been trained to reconstruct the original sentence from which the EAT was derived. Chapter 3 discussed it in the context of text transformation, but without going into detail on the implementation. Since the structure of EAT has been explained in the present chapter (Section 4.2.4), this section focuses on the technical aspects of EAT2seq.

EAT2seq uses LSTMs with two hidden layers for both the encoder and decoder (see Publication V for details). Interestingly, model performance did not improve when using bidirectional instead of unidirectional LSTMs. Reasons for this are unclear, but one possibility is that bidirectionality overcomplicated the encodings by adding too much contextual variation. Usually, bi-LSTMs are used with simpler data formats than EAT (e.g. word embeddings), which is a likely factor here.
The current implementation of EAT2seq is trained on sentences parsed (by Spacy) to the dependency grammar format, which is then mapped to EAT. The original sentences are used as the targets when training the text reconstruction. The decoder is a standard attention-LSTM. The first hidden decoder layer is initialized with the final encoding layer, and the decoder additionally uses attention across all input encodings to focus on the most relevant part of the (encoded) EAT-input. This setting does not differ from standard LSTM-based NMT networks. Instead of a decoder, it would also be possible to connect another DNN-model to the encoder to perform a different NLP task (e.g. text classification) based on EAT-input.

The trained EAT2seq model generates the sentence corresponding to the EAT-input. Since EAT clearly separates between grammatical features and argument structure, it is possible to use the model for grammatical transformation by changing the grammar without affecting the arguments. First, the original sentence is parsed (by Spacy) and mapped to EAT; second, the EAT is modified with respect to the relevant features; and finally, the modified EAT is run through EAT2seq to generate the target sentence. Figure 4.3a presents the EAT2seq pipeline.

EAT2seq thus provides a versatile technique for transforming specific properties of input sentences, without building this transformation itself into the encoder-decoder network. Since the encoder only takes the final
post-transformation EAT as input, it makes no difference to the success of the decoding stage which transformations it has gone through prior to this. Hence, any combination of transformations can be handled similarly by EAT2seq, provided that the final transformed EATs are sufficiently recognizable to the network. Crucially, the encoder-decoder network is trained only once, and the same network can be used for all transformations, to any direction. Publication V demonstrates EAT2seq’s success in text reconstruction and grammatical transformation.

In addition to the encoder-decoder-based EAT2seq, we also used the rule-based surface realization system SimpleNLG [99] for turning dependency-parsed input into English via EAT. This variant has more reliable performance than EAT2seq, but is more restricted in application and requires the full dependency-parsed sentence as input instead of only the EAT. Also, its reliance on SimpleNLG results in inheriting any of its deficiencies, and is a restriction against future application to languages that are not implemented in SimpleNLG. These complications notwithstanding, the EAT-SimpleNLG system depicted in Figure 4.3b had stronger overall performance on grammatical transformation than EAT2seq, making it a state-of-the-art solution to this task.

There are many possible further applications for EAT2seq and EAT-SimpleNLG. While Publication V concentrates on grammatical transformations, there is also no restriction against lexical transformations, which change one or more of the arguments in the E-, A-, or T-positions. Since the arguments are lemmatized and separated from their grammatical features, changing a lemma into another will automatically change the inflection appropriately for the new lemma (provided its grammatical status is the same as that of the original lemma). Lexical and grammatical transformations could also be combined freely.

Also, since it makes no difference to the encoder how the EAT has been derived, EAT2seq could be used to produce text from EAT-representations drawn from some other source than the syntactic parse of the original sentence. Original sentences are only required for training, since they provide the target outputs. In the test/application stage, only the EAT-input is needed, wherever it originates. For example, EAT could be mapped from another semantic representation like AMR, MRS, or DRT; with possible grammatical additions for original formats that lack such features (like AMR). Hence, EAT2seq is a broader framework for text generation than only a grammatical transformation scheme.

\[17\text{https://github.com/simplenlg/simplenlg}\]
4.4 Summary

Of the publications in this dissertation, this chapter has focused mostly on Publication V. Unlike in prior chapters, the themes were not directly related to security considerations, such as adversarial text or model evasion. Nevertheless, the application to text transformation connects EAT2seq/EAT-SimpleNLG to topics discussed in Publication III (Chapter 4; Section 3.2.3). The grammatical transformation module of ParChoice (Publication III) also used (an earlier version of) EAT2seq. Furthermore, semantic representation formats like EAT can be adopted for content classification, which has important repercussions for security and privacy (Chapter 2; Section 2.2.3).

While NLP applications for text generation have significantly developed over the last few years, detailed control of the output remains a challenge especially for DNN-based approaches. Tackling this matter requires careful attention to the input format. As discussed in Section 4.1.2, I believe there are reasons to doubt the ability of current DNNs to provide encodings that represent compositional meaning in a computationally tractable manner.

Among human-readable symbolic formats for semantic representation, there are multiple options to choose from depending on the task. Some variation of Neo-Davidsonian event semantics is standardly used by most frameworks, including AMR [19], MRS [72], and the DRT-based application Boxer [36]. EAT provides a novel alternative that aims to minimize the number of semantic relations without sacrificing either expressiveness or syntax-semantics transparency. Its simplicity allows it to be formalized into a list of fixed-size tuples, which is directly vectorizable and can be used to for a variety of NLP tasks.

Overall, EAT exhibits many properties desirable for a semantic representation format in NLP. While each of these can be individually found in prior semantic frameworks, they have so far not been combined as in EAT. This gives EAT certain important advantages in NLP applications, as reviewed in this chapter. The main properties are listed below:

- **simplicity**: very low number of semantic roles
- **expressivity**: detailed differentiation between meanings
- **theoretical basis**: explicitly motivated by linguistic theory
- **transparency**: direct derivability from the syntactic parse
- **linearity**: sequential rather than hierarchical structure
- **vectorizability**: direct translatability into a sequence of fixed-size vectors
5. Conclusions and future work

In the Introduction (Chapter 1) I divided the themes of this dissertation into three technical and three thematic classes. The technical classes are (i) text classification, (ii) text transformation, and (iii) text representation; and the thematic classes are (i) adversarial text, (ii) author profiling, and (iii) semantic representation (Table 1.1, repeated here as Table 5.1). Chapters 2–4 were based around the technical division, and this chapter summarizes the main takeaway messages from this perspective. I also make suggestions on what I consider to be worthwhile approaches to future research on related topics.

A major overarching theme is the comparison between DNNs and alternative NLP techniques. I suggest that many of our results are indicative of fundamental limits in DNNs, but there are also certain tasks where DNNs are clearly superior. I argue that only DNNs can (at the moment) produce sufficient output novelty required for many text generation tasks; but they fail to allow detailed human-control of the output. These results can be interpreted as a case for hybrid techniques that use both DNNs and symbolic NLP to maximize their respective benefits [103, 188].

Section 5.1 summarizes the main results across all five publications and Chapters 2–4. Section 5.2 then discusses the relation between DNNs and rule-based NLP in light of these results, and Section 5.3 summarizes the takeaway messages concerning security and privacy considerations. Finally, Section 5.4 suggests prospects for future work.

5.1 Main results

Sections 5.1.1–5.1.3 review the main high-level conclusions I draw from each of the nine cells in Table 5.1. Each section is allocated to a single technical class and divided by the three thematic classes.
Table 5.1. Classification of publications in this dissertation.

<table>
<thead>
<tr>
<th>Technical content</th>
<th>Thematic content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adversarial text</td>
</tr>
<tr>
<td></td>
<td>Author profiling</td>
</tr>
<tr>
<td></td>
<td>Semantic representation</td>
</tr>
<tr>
<td>Text classification</td>
<td>Publication I</td>
</tr>
<tr>
<td></td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication III</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
<tr>
<td>Text transformation</td>
<td>Publication I</td>
</tr>
<tr>
<td></td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
<tr>
<td>Text representation</td>
<td>Publication I</td>
</tr>
<tr>
<td></td>
<td>Publication II</td>
</tr>
<tr>
<td></td>
<td>Publication IV</td>
</tr>
</tbody>
</table>

5.1.1 Text classification

Our findings on text classification differed between content-based detection of adversarial text on the one hand, and author profiling on the other hand. However, a general observation can be made that the main predictor of classification performance was training data rather than model architecture. I also consider parallel corpus construction from EAT to be a variant of text classification in a broad sense, as it involves assigning texts to classes based on semantic content and grammatical features.

Adversarial text
Success in classifying deception (Publication I) or hate speech (Publication II, Publication IV) varied significantly based on training data, whereas model architecture did not have a clear correlation with performance. Even though DNNs generally constitute the state-of-the-art in text classification, here traditional ML techniques often had similar or even better performance (Publication II, Publication IV). A likely reason for this is that the relevant features were basic enough to be detected by simpler ML techniques, and there was no major need for complex context-depedencies that DNNs could potentially capture. Simpler ML techniques also have certain benefits over DNNs, such as better human-interpretability and smaller computational overhead (Publication IV).

Author profiling
State-of-the-art stylometry techniques allow detecting authors effectively, especially among relatively small sets of candidates ($\sim \leq 20$), which is demonstrated by much prior work (Publication I) and further corroborated by our novel empirical results (Publication III). For other classes than author identity, the success of profiling obviously depends on the extent to which those classes actually exhibit systematic stylistic variation. To the extent they do, stylometry techniques can be expected to detect them to a
significant degree. Our results point toward stylistic markers being more reliable for individual authors than for larger groups like gender or age (Publication III). Prior surveys [210, 278], our survey (Publication I), or our empirical results (Publication III) do not justify any general preference between DNNs and simpler ML techniques in stylometry.

**Semantic representation**
Generating parallel corpora via EAT (Publication V) can be treated as text classification of a very different kind than the other, ML-based classification tasks. Here, sentences are allocated to numerous classes based on sharing argument structure, and then sub-classified based on grammatical features within these argument structure classes. Reliably retrieving such information from sentences requires representations that are simultaneously able to contain all the relevant properties, as well as sufficiently simple to allow their feasible detection. EAT combines these properties.

### 5.1.2 Text transformation

Some text transformation tasks demand detailed control of high-level output properties, such as semantic content. Other tasks are more lenient in allowing changes, and only require e.g. thematic similarity to the original text. The three text transformation tasks studied in this dissertation were **model evasion** (Publication II, Publication III), **data augmentation** (Publication IV), and **grammatical transformation** (Publication V). Each has different challenges, leading to partly diverging recommendations on optimal techniques.

**Adversarial text**
Adversarial examples for text classification should fool the ML classifier but retain properties relevant for human interpretation. We showed that hate speech classifiers are easy to evade via simple text transformation attacks that do not require access to the targeted classifier (Publication II). These are based on (i) changing token identities (typos, whitespace injection/deletion), and (ii) adding unrelated innocuous material. While the first class of attacks can partly be mitigated by appropriate pre-processing (e.g. spell-checking), the second is a more fundamental problem for any system that conducts detection via classification. The presence of material indicative of another class will inappropriately take prediction probability toward that class, effectively hiding the presence of the relevant material (hate speech in our case). **Adversarial training** was partly able to mitigate the attacks; and similar but more elaborative techniques for artificial **data augmentation** were successful at improving hate speech classification on very small training sets (Publication IV).
Author profiling
Using style transfer to evade author profiling can be conducted with three main techniques. Iterative language translation via a MT system is the easiest solution, but very limited and unreliable (Publication I). Rule-based techniques allow user control of the changes, but have so far been narrow in scope. Encoder-decoder networks are the most popular among recent proposals, but do not exhibit sufficient semantic retainment, especially when trained on small datasets (Publication III). By using a combinatorial approach to paraphrase replacement with multiple modules, our ParChoice technique reached state-of-the-art style transfer performance, with significantly better semantic retainment than any of the five baseline techniques compared against (Publication III). However, like with hate speech detection (see above), adversarial training was usually able to mitigate model evasion.

Semantic representation
By separating argument structure from grammatical features (and lemmatizing words), EAT allows easy modification of some features while keeping everything else intact. EAT can be mapped to English via an encoder-decoder network or a rule-based surface realizer, which allows effective grammatical transformation to any direction with no task-specific training (Publication V). The same approach could also be broadened for lexical transformation without changing the model architecture or re-training.

5.1.3 Text representation
The problem of human-interpretability has recurred in many parts of this dissertation. Character or (sub)word features commonly used with DNNs tend to have only very indirect relations to properties relevant to most NLP tasks on the level of human understanding. Also, since intermediate encodings produced by encoder DNNs are not human-readable at all, it is impossible to say directly what information they actually contain, apart from indirect empirical evaluation based on experimental performance. These considerations result in fundamental challenges for user control of generated text, as well as information retrieval based on detailed linguistic content. Both are important aspects of NLP, and motivate the use of human-readable semantic representation formats together with DNN-generated embeddings or encondings, optimizing the benefits of each technique depending on the task.

Adversarial text
Relevant features for deception differ markedly between studies, and no general deception-markers emerge (Publication I). Our comparative experiments across hate speech datasets and classifiers are also indicative of
similar problem in this domain (Publication II). These observations show
that more work should be focused on understanding what features the
classifiers actually use, instead of only relying on performance measure-
ments like accuracy or F1-score. Using high-level human-readable input
features and simple ML techniques allows interpreting empirical results
better than using DNNs trained on characters or (sub)words. Hence, at
least for furthering our understanding of model performance, the former
approaches remain important. On the other hand, while character and
subword features have poor semantic interpretability in comparison to
higher-level features, they manifested superior overall performance in hate
speech detection (Publication II, Publication IV). A likely reason for this
is that similar words (e.g. different inflections of the same lemma) will
overlap more when using features below the word-level.

Author profiling
Stylistic classification has been conducted with a variety of feature
combinations, ranging from simple character- or word frequencies to hand-
crafted feature sets like Writeprints [312]. While any general consensus
on the most optimal features is lacking, character n-grams have shown
systematically good performance (Publication I). One major challenge
is differentiating genuinely stylistic features from those that are more
informative of semantic content, which can obviously vary between authors
more than writing style. Drawing this distinction appropriately (to the
extent it is possible at all) requires incorporating linguistic information
to the feature construction, and/or ensuring sufficient thematic overlap
between authors in the training set. The latter puts high restrictions on
the training set, and is not feasible in many cases. Therefore, linguistic
analysis continues to have a vital role in stylometry research.

Semantic representation
Linguistically motivated approaches to formal semantics in NLP are many-
fold, and Chapter 4 briefly covered three dominant frameworks: DRT
[140], AMR [19], and MRS [72]. Compared to these, the EAT-format
combines a much more austere structure with a transparent syntax-
semantics mapping, while still maintaining a high level of semantic
expressiveness (Publication V). Its use of the conjunctivist framework
[129, 231, 232, 233, 234, 235] makes possible the restriction of semantic
roles to the absolute minimum (three), which uniquely allows discarding
semantic metapredicates and instead using positional encoding for
expressing the roles.
5.2 DNNs vs. rule-based NLP

A major overarching theme in this dissertation has been the relationship between DNN-based NLP techniques and more traditional alternatives, such as simpler ML architectures for text classification or rule-based techniques for text transformation. I draw the general conclusion that their benefits and drawbacks are largely complementary.

In text classification, DNNs worked well with large training datasets, but with smaller datasets their performance did not systematically exceed that of simpler ML architectures. DNNs have the potential for capturing complex context-dependencies between features, abstracted at later hidden layers from original input tokens (characters or (sub)words). However, nothing compels a DNN to use complex feature relations instead of relying on more surface-level markers of the target class, such as simply the presence of a particular word or character n-gram. For example, hate speech detection systems are sensitive to curse words regardless of the context in which they appear (Publication II).

In text generation, DNNs are effective at producing synthetic text with high novelty. This makes them useful for tasks like data augmentation, where GPT-2 had the most success in our experiments (Publication IV). Here, more semantically restrictive techniques (e.g. synonym or paraphrase replacement) performed systematically worse. On the other hand, DNNs are less reliable when output features need to be tightly controlled. Style transfer is a clear example of this, and our results illustrate the problems of DNN-based encoder-decoder networks in retaining the original meaning to a sufficient degree (Publication III). While rule-based techniques are less versatile and more rigid in application, they also allow more detailed control of the output text.

In text representation, the distinction between tasks that are challenging for DNNs and symbolic/rule-based approaches largely coincides with the distinction between word-level semantics and the semantics of complex syntactic structures. Symbolic semantic representation formats tend to have little to say about the differences between (grammatically similar) words; whereas DNN-produced embeddings can effectively store distributional information about words. However, employing embeddings to represent the meaning of complex expressions is highly non-trivial, which motivates the use of symbolic representation formats. Additionally, while similarity among embeddings correlates with various kinds of semantic similarity, it does not reliably track the subtype of similarity, such as synonymy, antonymy, co-hyponymy, etc. (Chapter 4; Section 4.1.2). Rule-based NLP thus continues to hold its place in representing semantic structure, as well as higher-grained taxonomies of semantic relations.

I emphasize that none of the considerations reviewed here should be read as opposition to the use of DNNs. Instead, they concern the rela-
Conclusions and future work

The relationship between DNNs and their main alternatives, in particular symbolic rule-based NLP. An initially tempting analysis would be that DNNs are doing similar operations as rule-based techniques, but learning them from data instead of requiring them to be manually pre-programmed. Another possibility is that DNNs and rule-based techniques conduct different kinds of operations. If my arguments in this section are on the right track, they contribute to a case for the latter position, along with other work favoring similar conclusions in other domains [146, 159, 188].

5.3 Security and privacy considerations

Apart from Publication V, all publications of this dissertation directly concern security-relevant scenarios: classifying adversarial text (Publication I, Publication II, Publication IV), author identification or profiling (Publication I, Publication III), and model evasion (Publication II, Publication III) from both an attacker’s and a defender’s perspective. In this section I draw some general conclusions on the security implications of NLP, based on our results. These can be divided between two main topics: (i) NLP-based assets for enhancing security; and (ii) security and privacy threats posed by NLP. While both exist, I argue that the threats are more compelling.

The success of text classification on security-relevant tasks like the detection of deception or hate speech is notable but far from perfect. The most realistic application of automatic classification is as an initial filter for further, more detailed processing involving human evaluation. A high recall is especially important here. Since classifiers are likely to get caught on proxy-features that are not actually constitutive of the relevant class (Publication II), it is especially relevant to make sure that the training set contains enough lexical and grammatical variation. At least with very small training sets, automatic augmentation via synthetic data can significantly increase recall (Publication IV).

The flip-side to text classification is its use for censorship, and automatic recognition of material that leads to penalizing the authors. The better content-based text classification can work in tasks like hate speech detection, the better it can also be used for such coercive purposes. Furthermore, the availability of fine-grained semantic representation formats like EAT (Publication V) could be used to aid in the detection of very detailed semantic information, such as support or opposition with respect to particular viewpoints. This obviously raises a variety of concerns related to censorship and freedom of expression.

So-called “deception detection” is likely to focus on content-based features rather than semantically neutral stylistic properties (Publication I), which becomes increasingly relevant in the detection of fake news [214, 227]. Automatic techniques for fake news detection are almost certain to focus on...
Conclusions and future work

lexical content rather than deception-indicative stylistic features (for which we argued there to be no evidence). Fake news is also not always deceptive, since it might be sincerely believed by the author. Such complications again bring forward the intimate connection between content moderation and censorship – an issue with obvious political significance.

Additional problems for adversarial text detection are evasion techniques using text transformation. As Publication II demonstrated, these can be easily devised, and some make use of fundamental deficiencies in standard detection frameworks. In particular, a classifier can be brought to make the wrong prediction simply by burying the original text under irrelevant text that is indicative of the opposing class. Unlike some other attacks in Publication II, this is not possible to mitigate by data pre-processing.

Adversarial training is a useful mitigation technique that should be applied to models by updating their training data with adversarial examples. However, it can have a negative effect on classification performance on non-adversarial data. Also, the range of possible evasion techniques is open-ended, and attacks can be updated for new classifiers. For example, BERT [81] has been shown to be vulnerable to simple text transformation attacks [173]. This is in line with our observation that model architecture had no particular connection with susceptibility to evasion (Publication II).

In the application of stylometry, the security benefits of author identification include e.g. criminal detection [7] and identity proving in legal contexts [136]. However, its adversarial use remains a prominent privacy concern. Author deanonymization or profiling is a major privacy risk to authors who want to retain anonymity (Publication I). While the distinction between genuinely stylistic and content-based features is unclear (Section 5.1.3), successful author classification remains a privacy breach regardless of which it is based on. Even across a large number of candidate authors, stylometry can be used to limit the set of potential candidates to sufficiently small for further (manual or automatic) analysis.

Even though we were able to significantly improve on the state-of-the-art in combining writing style transfer with semantic retainment (Publication III), automatic means for evading stylometry are severely limited. Completely semantically equivalent paraphrases are relatively rare, resulting in at least some semantic changes often being the price of deanonymization. The most realistic application of automatic style transfer is to use it as a first stage in a pipeline that includes manual human involvement for e.g. proofreading and correcting inappropriate changes. This semi-automatic strategy has been adopted in some prior systems [77, 138, 193]. Manually proofread transformations by ParChoice were still able to take classification accuracy down significantly (Publication III), which indicates the feasibility of incorporating ParChoice into a semi-automatic framework.

Overall, the development of NLP systems has potential security benefits, but brings about even more pertinent security and privacy risks.
Content-based text classification is not sufficiently trustworthy to be relied on as means to deal with high-level problems like deception or hate speech, and model evasion is possible even against complex model architectures. Furthermore, the question of which texts should be allocated to which classes has obvious political relevance that cannot be solved by NLP alone, because it involves assessing the correctness of the ground-truth labeling itself. Privacy concerns related to author profiling are not sufficiently mitigated by automatic style transfer techniques, although they can be effective in a limited set of use cases (e.g. anonymous blogs or discussion forums). The problems are likely to increase along with the size and availability of datasets and large pre-trained models usable for transfer learning, which can improve model performance especially on small training sets [81, 133, 169, 175, 264].

5.4 Summary and future work

This dissertation has concerned three main technical topics and three main thematic topics within NLP, listed in Table 5.1. Publication V presents a generic technique for representing semantic content; and the other four publications relate directly to security and privacy considerations. Overall, the ongoing development of NLP technologies is likely to exert increasing privacy risks related to text classification by content or authorship. These techniques also have benign uses; but it is not possible to only have the benefits without the costs.

Furthermore, what constitutes “benign” use is a highly political question, not to be answered by technical considerations alone. Understanding NLP systems as such does not yet tell us whether original “ground-truth” labels of training sets are appropriate, or how the trained models should be used. In a recent review on resources for hate speech detection, Poletto et al. [236] cover significant discrepancies between studies concerning the proposed definitions of the relevant classes, labeling schemes, annotator guidance, and inter-annotator agreement. Accounting for such challenges requires approaching hate speech as a social, psychological, and ethical concern extending beyond NLP [42, 51].

Deliberate model evasion is feasible for mitigating classification, both from a defender’s and an attacker’s perspective. However, fully automatic text transformation with tight requirements on the output’s semantic content (e.g. style transfer) is challenging especially for DNN-based approaches (Publication III). Rule-based NLP as well as manual oversight for the transformation process are still required for ensuring sufficient semantic retention; let alone other considerations related to pragmatic aspects, stylistic fluency, etc.

Many observations made in this dissertation point to the importance of
Conclusions and future work

understanding the basis on which classifiers actually yield their outputs, as opposed to remaining fully agnostic about such questions and trusting their judgement without scepticism. A model might give a clearly above random success rate for e.g. deception detection, but still be unreliable if applied to different kinds of deceptive texts (Publication I). Hate speech classifiers are likely to get caught on proxy-features that can also be present in non-hateful speech; and conversely be vulnerable to less conventional forms of hate speech (Publication II).

One way to remedy such problems is to expand training sets with more variable examples, which is also feasible with synthetic data via text transformation (Publication IV). Another way is to improve our understanding of the most relevant features the models use. This can involve simpler ML techniques that have human-interpretable feature weights (e.g. logistic regression), or indirect means of assessing input feature relevance in DNNs. A common approach to the latter is attention visualization [285, 290]; although the ability of attention to yield genuine explanations of model behavior has also been put to question [294].

Focusing on model interpretation is thus crucial in the further development of text classification systems, especially for semantically complex target classes like deception or hate speech. In addition to model architecture, the choice of input features is highly relevant here. More semantically elaborate representation formats like EAT can allow detailed classification beyond the range of standard techniques (Publication V). Future work should advance the interaction between theory-driven research on semantic representation and data-driven NLP applications.

Many of our results speak in favor of the complementary nature of DNNs and rule-based techniques (Section 5.2), suggesting their integration in hybrid NLP systems [103]. Our main example of a hybrid NLP technique was EAT2seq (Publication V), which combines rule-based semantic parsing with NMT-based text generation. Such hybrid approaches are modular: the techniques are applied in succession at different parts of the pipeline. Beyond these, there are also alternatives that build symbolic processes within DNNs themselves. Adding external memories to DNNs via architectures like Neural Turing Machines [108] has shown benefit in certain tasks that have been difficult for standard DNNs, such as the extrapolation of operations outside the training domain [146, 159]. Recently, another approach has been developed that assigns nodes to components of logical formulae and performs inference in a manner that respects compositional semantics [256]. Inclusion of such techniques is an important venue for future research in NLP.

As discussed in Chapter 4 (Section 4.1.2), one strand of NLP research aims to model compositional semantics via tensor operations [22, 23, 78, 86, 109, 202, 203]. While such approaches have not been on the forefront in mainstream NLP, they have clear relevance for many themes raised in this
Conclusions and future work

dissertation. Typically, they rely on modelling Fregean semantic adicity as tensor dimensionality. For example, Baroni and Zamparelli [23] assign nouns to vectors and adjectives to matrices, such that matrix multiplication results in a vector representing the adjective-noun combination. Similar techniques have been extended to other basic linguistic constructions, such as subject-verb-object triplets [78], prepositional phrases [22], and logical connectives [109]. EAT greatly reduces the range of semantic types and combinatorial operations compared to standard Montagovian semantics, which increases its aptitude for possible future unification with such tensor-based approaches to semantic computation. However, the entanglement problem discussed in Section 4.1.2 of Chapter 4 is present here as well: even if a complex semantic representation is computed from certain arguments, this does not yet ensure that these arguments are present in it as computationally accessible constituents.

Finally, on a broader note, the question of whether NLP systems actually represent meaning has recently increased in prevalence [27, 85, 192, 196, 213]. Concerns have especially been raised about models like BERT excessively relying on surface-level features [192, 196, 213]. Bender and Koller [27] bring up a more foundational issue reminiscent of prior discussions in the philosophy of language [120, 167, 269]: can meaning ever be captured by formal linguistic properties, or does it require a relation between language and some non-linguistic aspect of the world? Opting for the latter view, Bender and Koller argue that it sets a fundamental limitation on any NLP system claimed to capture meaning by being trained on linguistic data alone. While such deep theoretical matters are largely beyond the reach of this dissertation, the results outlined in this concluding chapter illustrate that more thorough consideration is needed to draw conclusions about what NLP systems are actually doing when they engage in tasks that we – as humans – interpret as semantic.
References


References


[33] George Boolos. To be is to be the value of a variable (or the values of some variables). *Journal of Philosophy*, 81:430–450, 1984.


References


References


References


References


References


This doctoral dissertation investigates natural language processing (NLP) in a variety of settings, with a focus on tasks relevant for information security and privacy. The increasing performance and availability of NLP techniques has led to their broad application for text classification and generation. While this has potential for detecting security breaches such as deception or hate speech, it also allows malicious use for author profiling, model evasion, and beyond. Properly understanding such repercussions of NLP requires thorough comparison of model architectures, datasets, and methods for input representation. The dissertation contributes to this line of research from multiple angles, with a focus on using NLP to detect security- or privacy-sensitive information on the one hand, and the evasion of such detection on the other hand. It additionally proposes a novel framework for computational semantics, and discusses the relationship between deep learning and rule-based NLP.