BEng Individual Project

Imperial College London

Department of Computing

ADA-X: A System for Domain-Independent Feature-Based Product Review Aggregation

Supervisor:
Prof. Francesca Toni

Author:
Joel Oksanen

Second Marker:
Prof. Alessandra Russo

June 15, 2020

Abstract

When looking to buy a product, consumers often look for written reviews to gain insight into its various features. However, many reviews are cluttered with irrelevant information, which makes reading through them an arduous task. Some systems have been proposed to alleviate this task in specific review domains such as film reviews, but a general solution remains to be implemented. We present ADA-X, a general-domain review aggregation system that is able to provide feature-based explanations for any product's reviews via a conversational interface. The generality of ADA-X is enabled by a novel domain-independent ontology extraction method using the state-of-the-art language model BERT, which we have shown to obtain product ontologies with an F1-score multiple times higher than the state-of-the-art general knowledge networks of ConceptNet and WordNet.

ADA-X has been demonstrated to be capable of providing conversational feature-based review explanations for any Amazon product, and its methodology could be extended to any review domain with sufficient amounts of data. User evaluation has demonstrated that the explanations provided by ADA-X improve the user experience of product review inspection.

Acknowledgements

First of all, I would like to express my deepest gratitude to Professor Francesca Toni, my supervisor, for allowing me to extend upon her own work, and for her invaluable guidance through the course of this project.

I would also like to thank Ran Ji, Noora Kovalainen, Jenny Pan, and Nina Zhao for helping me evaluate the various aspects of ADA-X.

Contents

1 Introduction **6**
- 1.1 Motivations . . . 6
- 1.2 Objectives . . . 6
- 1.3 Contributions . . . 7

2 Background **9**
- 2.1 Argumentative Dialogical Agent . . . 9
  - 2.1.1 Feature-based representation . . . 10
  - 2.1.2 Review aggregation . . . 10
  - 2.1.3 Quantitative Bipolar Argumentation Framework . . . 12
  - 2.1.4 Gradual semantics . . . 12
  - 2.1.5 Dialogical explanations . . . 13
  - 2.1.6 ADA for Amazon reviews . . . 14
- 2.2 Feature-level sentiment analysis . . . 16
  - 2.2.1 Ontology extraction . . . 16
  - 2.2.2 Feature-dependent sentiment analysis . . . 17
- 2.3 BERT . . . 18
  - 2.3.1 Tokeniser . . . 18
  - 2.3.2 Transformer network . . . 19
  - 2.3.3 Fine-tuning BERT . . . 20
- 2.4 Conversational systems . . . 20
  - 2.4.1 Botplications . . . 20
  - 2.4.2 Quantised Dialogue . . . 21
  - 2.4.3 Evaluation using the RRIMS properties . . . 21

3 Ontology extraction **22**
- 3.1 Implementation . . . 22
  - 3.1.1 Annotation of training data for masked BERT . . . 22
  - 3.1.2 Noun extraction . . . 25
  - 3.1.3 Feature extraction . . . 25
  - 3.1.4 Synonym extraction . . . 26
  - 3.1.5 Ontology extraction . . . 27
- 3.2 Evaluation . . . 29
  - 3.2.1 Ontology evaluation . . . 29
  - 3.2.2 Generalisation evaluation . . . 31

4 Feature-dependent sentiment analysis **33**
- 4.1 Availability of data . . . 33
- 4.2 Implementation . . . 34
  - 4.2.1 TD-BERT . . . 34
- 4.3 Evaluation . . . 35
  - 4.3.1 Annotation of Amazon sentiment analysis data . . . 35
  - 4.3.2 Results . . . 35

5 ADA-X System **37**
5.1 Architecture 37
5.1.1 Back-end 37
5.1.2 iOS Botplication 39
5.2 Evaluation 41
5.2.1 Performance 41
5.2.2 User evaluation 42

6 Conclusion **45**
6.1 Future work 45
6.1.1 Extending the conversational aspect 45
6.1.2 Recommender system 46

A Extracted ontologies **47**

List of Figures

1.1 Ontology for a wrist watch 7
1.2 A conversation between the user and ADA-X on the iOS app about a wrist watch 8

2.1 ADA as proposed by Cocarascu et al. for the Rotten Tomatoes movie review domain 9
2.2 Feature-based representation for a digital camera 10
2.3 QBAF for a digital camera annotated with base scores τ and DF-QuAD strengths σ 12
2.4 Auxiliary functions for argumentation dialogue. 13
2.5 Overview of the BERT architecture 18
2.6 A single encoder 19

3.1 Entity annotator interface 23
3.2 BERT for feature extraction 25
3.3 BERT for relation extraction 27
3.4 Function for constructing the ontology tree 28
3.5 Accuracies on the w_e test set for masked BERT models trained with datasets $t_1 \dots t_5$ containing n categories of products. The accuracies of the models trained with in-domain data w_t are shown as dashed lines. 32

4.1 TD-BERT architecture 34
4.2 Entity sentiment annotator interface 35

5.1 The full ADA-X system architecture, exemplified for a camera product 38
5.2 Screenshots from the ADA front-end applications 40
5.3 The experiment set-up for each of the interfaces, with all product information such as titles and ratings hidden. 43

A.1 Full ontologies for *video game* 47
A.2 Full ontologies for *watch* 48
A.3 Full ontologies for *television* 49
A.4 Full ontologies for *necklace* 50
A.5 Full ontologies for *stand mixer* (*the term *stand mixer* returned no results in WordNet, so *mixer* was used instead) 51

List of Tables

3.1 Example training data for feature extraction 24
3.2 Example training data for relation extraction 24
3.3 Training data counts for ontology extraction 24
3.4 Number of extracted relations for the three ontology extraction methods 29
3.5 Precision scores for the three ontology extraction methods 29
3.6 Relative recall scores for the three ontology extraction methods 30
3.7 F1 scores for the three ontology extraction methods 30
3.8 Products included in each of the datasets 31

4.1 SemEval-2014 Task 4 data counts 33
4.2 Example of SemEval-2014 Task 4 data 33
4.3 Amazon sentiment analysis evaluation data counts 35
4.4 TD-BERT sentiment analysis evaluation results 36

5.1 Average computation times for the ontology extraction process with 30,000 reviews, extracting on average 18.4 features per ontology, with 0.448 synonyms per feature . 41
5.2 Average computation times for the QBAF extraction process with 1,000 reviews . . 41
5.3 Mean response times for the system's various interactive tasks 42
5.4 User interface trial results 44

Chapter 1

Introduction

In this chapter, we will first discuss the motivations behind the project and then specify its main objectives and contributions.

1.1 Motivations

People spend an ever growing share of their earnings online, from purchasing daily necessities on e-commerce sites such as Amazon to streaming movies on services such as Netflix. As the market shifts online, people's purchase decisions are increasingly based on product reviews either accompanying the products on e-commerce sites, or on specialised review websites, such as Rotten Tomatoes[1] for movies. These reviews can be written by fellow consumers who have purchased the product or by professional critics, such as in the latter example, but what unites most online review platforms is the massive number of individual reviews: a single wrist watch can have more than 20,000 reviews on Amazon[2]. As people cannot possibly go through all of the individual reviews, purchase decisions are often based on various kinds of review aggregations. The presentation of a review aggregation must be concise and intuitive in order to be effective, but it must also retain some nuances of the original reviews, so that consumers can understand *why* a product is considered good or bad and if the arguments align with their individual preferences.

Explanations of review aggregations are also relevant to e-commerce site recommender systems, as their recommendations are often based on existing consumer reviews. It has been shown that explanations can help to improve the overall acceptance of a recommender system [7], and therefore insightful explanations of review aggregations are important to their effectiveness.

1.2 Objectives

Some systems have already been proposed to improve explanations for review aggregations, a few of which are discussed in Chapter 2. One such system is what is called an Argumentative Dialogical Agent (ADA), proposed by Cocarascu et al. in [3], which has been explored in the movie and hotel review domains of Rotten Tomatoes [3] and Trip Advisor[3] [2], respectively. ADA provides the user with insight about reviews for a particular product through conversation centred around the product's features. Cocarascu et al. provide the following example of a conversation about the film *The Post*:

User:	*Why was The Post highly rated?*
ADA:	*The movie was highly rated because the **acting** was great, although the **writing** was poor.*
User:	*Why was the acting considered to be great?*
ADA:	*Its acting was considered to be great because **Meryl Streep** was great.*
User:	*What did critics say about Meryl Streep being great?*
ADA:	*"...Streep's hesitations, rue, and ultimate valor are soul-deep..."*

[1]https://www.rottentomatoes.com/
[2]https://www.amazon.co.uk/Casio-Collection-Unisex-Adults-F-91W-1YER/dp/B000J34HN4
[3]https://www.tripadvisor.com/

In order to participate in such dialog, ADA has to have knowledge about the *ontology* of the product (i.e. *acting* is a sub-category of *film* and *Meryl Streep* is a sub-category of *acting*), as well as knowledge about the reviewers' sentiments towards the various aspects of the film. ADA obtains some of this knowledge from the review texts using *natural language processing* (NLP) methods. In the papers by Cocarascu et al., the ontologies for films and hotels are mostly given, although mining some aspects from the review texts was also experimented with. The opinions of reviewers were extracted from the review texts using *sentiment analysis* (SA) methods.

The objective of this project is to extend upon the work of Cocarascu et al. in order to design and implement a generalised ADA to provide explanations for any product reviews on Amazon. As Amazon contains reviews for products ranging from pet supplies to digital music and car parts, this involves taking a more unsupervised and domain-independent approach to the various NLP tasks within the agent, as well as considering the effects of such variance on the dialogue between the user and the agent. Furthermore, ADA's complex methodology has never before been implemented within a fully developed end-to-end system, presenting a considerable software engineering challenge.

1.3 Contributions

The main contributions of this project are threefold:

1. We propose and implement a novel method for domain-independent product ontology extraction, which allows us to build a *feature-based representation* of any product in an unsupervised manner, an example of which is shown in Figure 1.1 for wrist watches. The ontologies extracted with this method obtain significantly higher F-1 scores when compared to the widely used manually and autonomously annotated semantic networks WordNet and ConceptNet, respectively.

2. We implement a modified version of the method by Gao et al. [6] for feature-level sentiment analysis, which allows us to assess the reviewers' sentiment towards the various features of the product, whilst improving upon the accuracy of the phrase-level sentiment analysis in [3].

3. We implement ADA for the first time and from scratch as a fully developed system, with a backend agent including the aforementioned enhancements and a user-facing iOS chat application, which is exemplified for a wrist watch in Figure 1.2.

For both 1. and 2. we use BERT, a language model proposed by Devlin et al. [4], which has produced state-of-the-art results in a wide variety of different NLP tasks. We call the resulting system ADA-X for its ability to generalise for any product X.

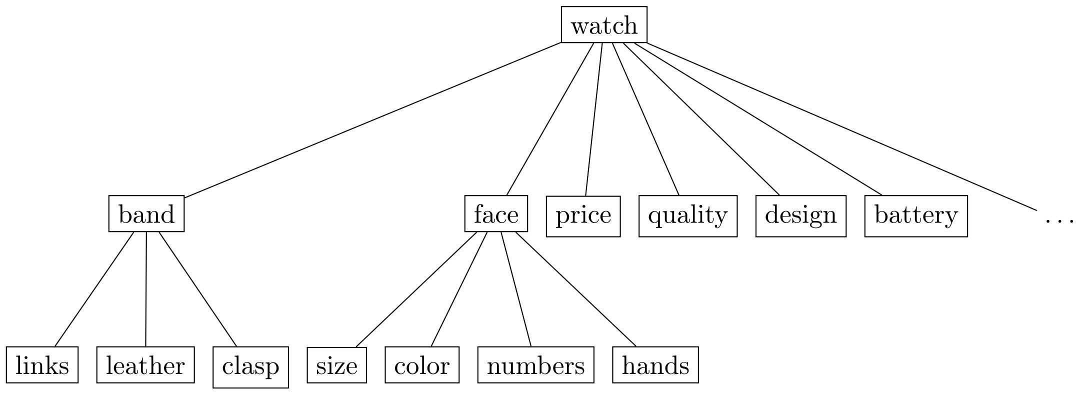
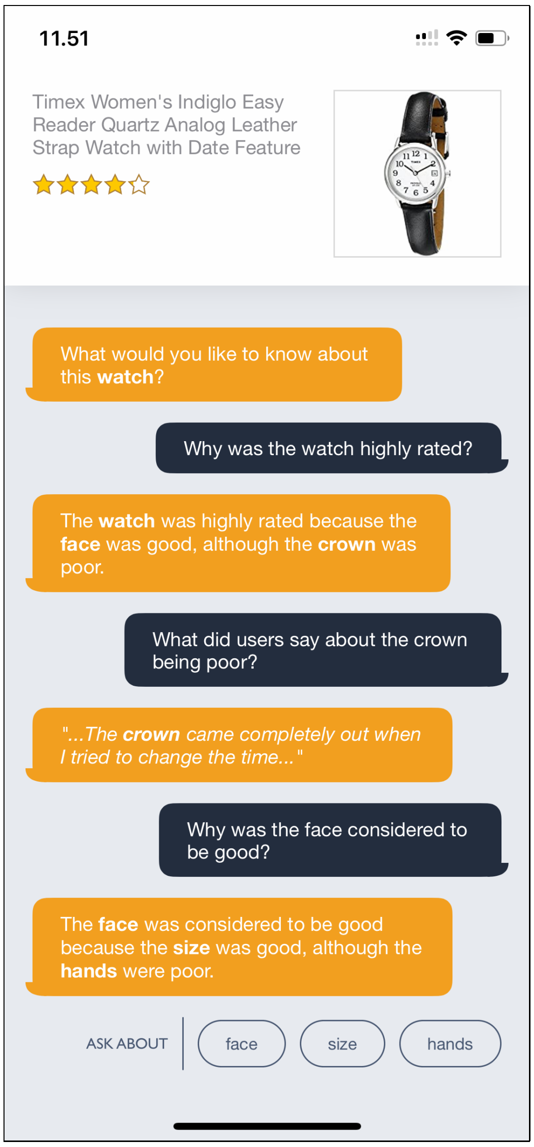


Figure 1.1: Ontology for a wrist watch


11.51

Timex Women's Indiglo Easy Reader Quartz Analog Leather Strap Watch with Date Feature

What would you like to know about this **watch**?

Why was the watch highly rated?

The **watch** was highly rated because the **face** was good, although the **crown** was poor.

What did users say about the crown being poor?

*"...The **crown** came completely out when I tried to change the time..."*

Why was the face considered to be good?

The **face** was considered to be good because the **size** was good, although the **hands** were poor.

Figure 1.2: A conversation between the user and ADA-X on the iOS app about a wrist watch

Chapter 2

Background

We begin this chapter by detailing the methodology of the ADA proposed by Cocarascu et al. [3]. We then evaluate the limitations of ADA in relation to Amazon reviews, and suggest extensions to address them. Then, we look into current research in the fields of ontology extraction and feature-dependent sentiment analysis in order to establish a basis for our enhancements to the agent. We conclude this chapter with a discussion about the current research on conversational systems.

2.1 Argumentative Dialogical Agent

ADA is designed around a *feature*-based conceptualisation of products. Rather than simply reporting the general sentiment of reviewers towards a product, ADA builds a more intricate understanding of *why* a product is (dis-)liked by inspecting the reviewers' sentiments towards its features. For example, rather than stating that a movie is mostly well-received, ADA can discern that reviewers appreciated its acting and themes, while its cinematography was found subpar.

Figure 2.1 presents an overview of how ADA works in the movie review domain. It takes as input reviews snippets for a particular movie, and outputs a dialectical strength measure for the movie along with dialogical explanations for the proposed strength. The ADA pipeline can be divided into three main functions:

1. ADA uses Natural Language Processing (NLP) methods to mine a *review aggregation* (RA) for the movie, which consists of a set of *votes* $\mathcal{V}$ expressing the reviewers' sentiments on the movie and its features.

2. The review aggregation is used to generate a Quantitative Bipolar Argumentation Framework (QBAF), which summarises the reviewers' general consensus on each of the features.

3. *Gradual semantics* are applied to the QBAF to calculate the strength measure and provide dialogical explanations.

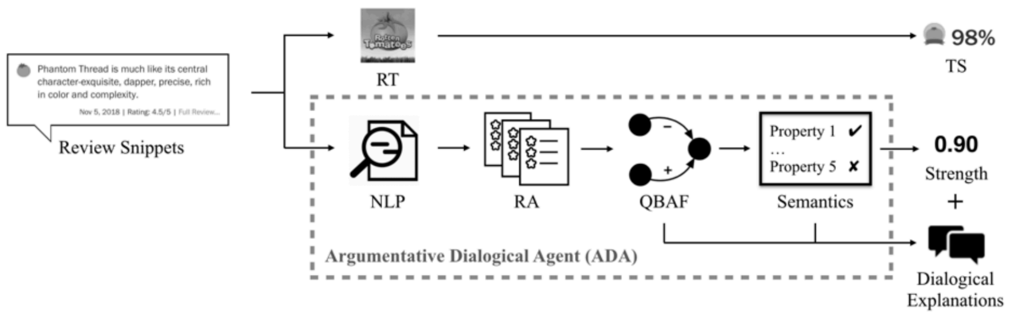

Phantom Thread is much like its central character-exquisite, dapper, precise, rich in color and complexity.
Nov 5, 2018 | Rating: 4.5/5 | Full Review...

Figure 2.1: ADA as proposed by Cocarascu et al. for the Rotten Tomatoes movie review domain

Considering a more generic domain, we will from now on refer to the object under review as a *product* p. Furthermore, as reviewers can have opinions on the product as a whole or the product's *features* $\mathcal{F}$, we refer to $\mathcal{A} = \{p\} \cup \mathcal{F}$ as the set of *arguments* upon which votes can act.

In the following sections, we will go through the different aspects of ADA in more detail. We will conclude section 2.1 by discussing the limitations of ADA with specific regards to Amazon reviews.

2.1.1 Feature-based representation

ADA is designed around a *feature-based characterisation* of products:

Definition 2.1 (Feature-based characterisation). A feature-based characterisation of a product p is a set $\mathcal{F}$ of features with sub-features $\mathcal{F}' \subset \mathcal{F}$ such that each $f' \in \mathcal{F}'$ has a unique parent $p(f') \in \mathcal{F}$; for any $f \in \mathcal{F} \backslash \mathcal{F}'$, we define $p(f) = p$.

This feature-based characterisation can be *predetermined*, i.e. obtained from metadata. For example, for a digital camera p, we could use metadata to obtain $\mathcal{F} = \{f_I, f_L, f_B\}$, where f_I is *image quality*, f_L is *lens*, and f_B is *battery*, as these are features that are present in most, if not all, digital cameras. Features can also be *mined* from the reviews themselves using more sophisticated methods, such as the semantic network ConceptNet[1] to identify related terms to either the product or its known features, in order to obtain further sub-features. For our camera example, we could mine the sub-features f'_{L1} for *zoom* and f'_{L2} for *autofocus*. Figure 2.2 shows the full feature-based representation for our running example.

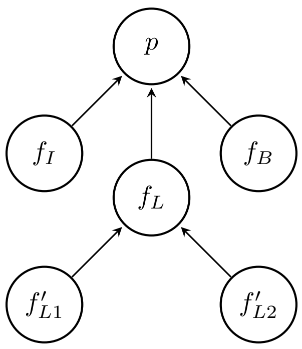


Figure 2.2: Feature-based representation for a digital camera

2.1.2 Review aggregation

ADA mines votes for and against the arguments in $\mathcal{A}$ from the review snippets, which together form a review aggregation for the product:

Definition 2.2 (Review aggregation). A review aggregation for product p is a tuple $\mathcal{R}(p) = \langle \mathcal{U}, \mathcal{V} \rangle$ where $\mathcal{U}$ is a finite, non-empty set of users (reviewers) and $\mathcal{V} : \mathcal{U} \times \mathcal{A} \rightarrow \{-, +\}$ is a partial function, with $\mathcal{V}(u, \alpha)$ representing the vote of user u on argument α.

The mining is performed on each review snippet individually, and follows a three-step process of *tokenisation*, *argument detection*, and *sentiment analysis* detailed below.

Tokenisation The review snippet is tokenised into sentxences, which are further split into phrases at specific keywords (*but*, *although*, *though*, *otherwise*, *however*, *unless*, *whereas*, *despite*). Each phrase can then possibly constitute a negative or a positive vote for one or multiple arguments.

[1]http://conceptnet.io/

Argument detection ADA determines the arguments on which a vote acts using a *glossary* $\mathcal{G}$ of words related to each argument. For our earlier digital camera example, the glossary could be as follows:

$$\begin{aligned}
\mathcal{G}(p) &= \{camera, device, product\};\\
\mathcal{G}(f_I) &= \{image, picture, photo\};\\
\mathcal{G}(f_L) &= \{lens\};\\
\mathcal{G}(f_B) &= \{battery\};\\
\mathcal{G}(f'_{L1}) &= \{zoom\};\\
\mathcal{G}(f'_{L2}) &= \{autofocus\}.
\end{aligned}$$

A phrase constitutes a vote towards an argument α if the phrase contains a word from $\mathcal{G}(\alpha)$. Sub-features take precedence over their parents, so for example a phrase with words from both $\mathcal{G}(f)$ and $\mathcal{G}(p)$ will constitute a single vote for f. A phrase that contains words corresponding to multiple unrelated features results in a vote for each of the corresponding features.

Sentiment analysis ADA uses NLP methods to determine whether the vote is for or against the argument(s). In their paper, Cocarascu et al. compared two different NLP methods for this purpose, Sentiment Analysis (SA) and Argument Mining (AM), finding that SA performed slightly better than AM. Furthermore, SA is a more established field of research than AM, which supports our decision to use SA for this project. The ADA for Rotten Tomatoes used an off-the-shelf classifier on movie reviews for its SA, which calculates a sentiment polarity for the phrase in $[-1, 1]$, with -1 and 1 denoting universally negative and positive sentiments respectively. A phrase with a highly positive (negative) polarity forms a vote for (against) the argument(s), while a phrase with an absolute polarity less than a pre-set polarity threshold of 0.6 is filtered out as neutral.

To illustrate review aggregation, consider this digital camera review from user u:

The picture quality was great, although the zoom wasn't as good as advertised.

ADA extracts two phrases, p_1: *The image quality was great* and p_2: *the zoom wasn't as good as advertised*. ADA then connects p_1 with f_I and p_2 with f_B as *picture* $\in G(f_I)$ and *zoom* $\in G(f'_{L1})$. ADA then obtains the sentiment polarities $(p_1, 0.863)$ and $(p_2, -0.629)$. As the polarity for both surpass the threshold of ± 0.6, ADA assigns two votes of $\mathcal{V}(u, f_I) = +$ and $\mathcal{V}(u, f'_{L1}) = -$ for the phrase.

Augmented review aggregation

Online reviews are often brief: Amazon reviews have a median length of just 82 words according to a 2013 study [15]. This can result in sparsely populated review aggregations. To fix this issue, the review aggregation can be augmented as follows:

Definition 2.3 (Augmented review aggregation). Given a review aggregation $\mathcal{R}_0(p) = \langle \mathcal{U}, \mathcal{V}_0 \rangle$, an augmented review aggregation $\mathcal{R}(p) = \langle \mathcal{U}, \mathcal{V} \rangle$ is such that for all $u \in \mathcal{U}$ and $\alpha \in \mathcal{A}$:

- if $\mathcal{V}_0(u, \alpha)$ is defined then $\mathcal{V}(u, \alpha) = \mathcal{V}_0(u, \alpha)$; else
- if $|\{\beta \in \mathcal{A} | p(\beta) = \alpha \land \mathcal{V}_0(u, \beta) = +\}| > |\{\gamma \in \mathcal{A} | p(\gamma) = \alpha \land \mathcal{V}_0(u, \gamma) = -\}|$ then $\mathcal{V}(u, \alpha) = +$; else
- if $|\{\beta \in \mathcal{A} | p(\beta) = \alpha \land \mathcal{V}_0(u, \beta) = +\}| < |\{\gamma \in \mathcal{A} | p(\gamma) = \alpha \land \mathcal{V}_0(u, \gamma) = -\}|$ then $\mathcal{V}(u, \alpha) = -$; else
- $\mathcal{V}(u, \alpha)$ is undefined.

The augmentation populates the review aggregation by exploiting the parent relation of arguments and the fact that an argument can be seen as a sum of its sub-features. If an argument does not have a vote from a user u, but its sub-features have primarily positive (negative) votes from u, it will also gain a positive (negative) vote from u.

2.1.3 Quantitative Bipolar Argumentation Framework

ADA uses the review aggregation to generate a QBAF, which is a quadruple $\langle \mathcal{A}, \mathcal{L}^-, \mathcal{L}^+, \tau \rangle$ consisting of a set $\mathcal{A}$ of arguments, binary child-parent relations $\mathcal{L}^-$ (attack) and $\mathcal{L}^+$ (support) on $\mathcal{A}$, and a total function $\tau : \mathcal{A} \to [0,1]$ representing the *base score* of the arguments. It is defined as follows:

Definition 2.4 (QBAF). Let $\mathcal{R}(p) = \langle \mathcal{U}, \mathcal{V} \rangle$ be any (augmented) review aggregation for product p. For any argument $\alpha \in \mathcal{A}$, let $\mathcal{V}^+(\alpha) = |\{u \in \mathcal{U} | \mathcal{V}(u, \alpha) = +\}|$ and $\mathcal{V}^-(\alpha) = |\{u \in \mathcal{U} | \mathcal{V}(u, \alpha) = -\}|$ denote the number of votes for and against α. Then the QBAF corresponding to $\mathcal{R}(p)$ is $\langle \mathcal{A}, \mathcal{L}^-, \mathcal{L}^+, \tau \rangle$ such that:

$$\begin{aligned}
\mathcal{L}^- = &\{(\alpha, \beta) \in \mathcal{F}^2 | \beta = p(\alpha) \wedge \mathcal{V}^+(\beta) > \mathcal{V}^-(\beta) \wedge \mathcal{V}^+(\alpha) < \mathcal{V}^-(\alpha)\} \cup \\
&\{(\alpha, \beta) \in \mathcal{F}^2 | \beta = p(\alpha) \wedge \mathcal{V}^+(\beta) < \mathcal{V}^-(\beta) \wedge \mathcal{V}^+(\alpha) > \mathcal{V}^-(\alpha)\} \cup \\
&\{(\alpha, p) \in \mathcal{F} \times p | p = p(\alpha) \wedge \mathcal{V}^+(\alpha) < \mathcal{V}^-(\alpha)\}; \\
\mathcal{L}^+ = &\{(\alpha, \beta) \in \mathcal{F}^2 | \beta = p(\alpha) \wedge \mathcal{V}^+(\beta) > \mathcal{V}^-(\beta) \wedge \mathcal{V}^+(\alpha) > \mathcal{V}^-(\alpha)\} \cup \\
&\{(\alpha, \beta) \in \mathcal{F}^2 | \beta = p(\alpha) \wedge \mathcal{V}^+(\beta) < \mathcal{V}^-(\beta) \wedge \mathcal{V}^+(\alpha) < \mathcal{V}^-(\alpha)\} \cup \\
&\{(\alpha, p) \in \mathcal{F} \times p | p = p(\alpha) \wedge \mathcal{V}^+(\alpha) > \mathcal{V}^-(\alpha)\};
\end{aligned}$$

$$\tau(p) = 0.5 + 0.5 \cdot \frac{\mathcal{V}^+(p) - \mathcal{V}^-(p)}{|\mathcal{U}|} \quad \text{and} \quad \forall f \in \mathcal{F}, \tau(f) = \frac{|\mathcal{V}^+(f) - \mathcal{V}^-(f)|}{|\mathcal{U}|}.$$

The QBAF represents attack and support relations between the various arguments, as well as a base score for each argument. An attack in $\mathcal{L}^-$ is defined between a sub-feature with primarily positive (negative) votes and its parent feature with primarily negative (positive) votes, or from a feature with primarily negative votes towards the product p. A support in $\mathcal{L}^-$ is defined correspondingly between a sub-feature and its parent feature both with primarily positive (negative) votes, or from a feature with primarily positive votes towards the product p.

The base score of an argument is related to the general sentiment towards it, and is defined differently for the product and its features. For the product itself, a base score of 0.0 (1.0) represents a completely negative (positive) sentiment from every user, while a base score of 0.5 represents a completely neutral sentiment. For features, the base score represents the magnitude of the positive or negative sentiment towards it, with 0.0 representing a neutral sentiment and 1.0 representing a fully negative or positive sentiment.

Figure 2.3 shows an example of a QBAF for a digital camera, with the arrows representing attacks ($-$) and supports ($+$), and the base scores τ shown next to each argument.

2.1.4 Gradual semantics

Reviews often integrate a numerical user rating in addition to the review text, such as a rating out of 5 stars. The rating can be averaged across reviews to find a quantitative measure for the general sentiment towards the product. It is beneficial to calculate a similar objective measure based on the QBAF to compare with the average user rating, in order to evaluate the accuracy of ADA. To calculate such a measure, we need to define a gradual semantics for the QBAF.

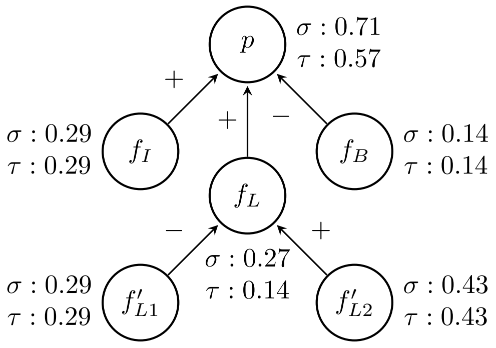


Figure 2.3: QBAF for a digital camera annotated with base scores τ and DF-QuAD strengths σ

Cocarascu et al. evaluated three different gradual semantics for this purpose: QuAD, DF-QuAD, and *Restricted Euler-based semantics* (REB). DF-QuAD performed best in comparison to the ground truth, while REB performed worst. However, REB satisfies the property of *strict monotonicity*, which was hypothesised to be 'more suitable for applications with a large number of attackers or supporters'. Since the most popular Amazon products have thousands of reviews as opposed to a few hundred reviews on the most popular movies on Rotten Tomatoes[2], we will use both DF-QuAD and REB in this project in order to evaluate this claim.

Let $\sigma(\alpha) \in [0,1]$ represent the *dialogical strength measure* for argument α. For a QBAF $\langle \mathcal{A}, \mathcal{L}^-, \mathcal{L}^+, \tau \rangle$ and argument $\alpha \in \mathcal{A}$, let $\mathcal{L}^-(\alpha) = \{\beta \in \mathcal{A} | (\beta, \alpha) \in \mathcal{L}^-\}$ be the set of attackers of α and $\mathcal{L}^+(\alpha) = \{\beta \in \mathcal{A} | (\beta, \alpha) \in \mathcal{L}^+\}$ be the set of supporters of α. Then DF-QuAD and REB are used to calculate σ as follows:

Definition 2.5 (DF-QuAD). For k arguments with strengths $\sigma_1, \sigma_2, \cdots, \sigma_k$, we define the aggregate strength measure

$$\mathcal{F}(\sigma_1, \sigma_2, \cdots, \sigma_k) = 1 - \prod_{i=1}^{k} (1 - \sigma_i).$$

Then for an argument $\alpha \in \mathcal{A}$ with n attackers with strengths $\sigma_{a1}, \sigma_{a2}, \cdots, \sigma_{an}$ and m supporters with strengths $\sigma_{s1}, \sigma_{s2}, \cdots, \sigma_{sm}$, where $\sigma_a = \mathcal{F}(\sigma_{a1}, \sigma_{a2}, \cdots, \sigma_{an})$ and $\sigma_s = \mathcal{F}(\sigma_{s1}, \sigma_{s2}, \cdots, \sigma_{sm})$:

- if $\sigma_a = \sigma_s$ then $\sigma(\alpha) = \tau(\alpha)$; else
- if $\sigma_a > \sigma_s$ then $\sigma(\alpha) = \tau(\alpha) - \tau(\alpha) \cdot |\sigma_s - \sigma_a|$; else
- $\sigma(\alpha) = \tau(\alpha) + (1 - \tau(\alpha)) \cdot |\sigma_s - \sigma_a|$.

Definition 2.6 (REB). For any argument $\alpha \in \mathcal{A}$:

$$\sigma(\alpha) = 1 - \frac{1 - \tau(\alpha)^2}{1 + \tau(\alpha) \cdot e^E}$$

where $E = \sum_{\beta \in \mathcal{L}^+(\alpha)} \sigma(\beta) - \sum_{\gamma \in \mathcal{L}^-(\alpha)} \sigma(\gamma)$.

For a product p, a low (high) $\sigma(p)$ signifies a negative (positive) overall sentiment towards the product and its features, while $\sigma(p) = 0.5$ signifies a neutral sentiment. Therefore, $\sigma(p)$ can be compared with the average user rating for p in order to evaluate the accuracy of ADA.

The QBAF in figure 2.3 is annotated with DF-QuAD strengths for its arguments. Since $p(\sigma) = 0.71$, users have an overall positive sentiment towards the camera.

2.1.5 Dialogical explanations

ADA can generate dialogical explanations for arguments based on the extracted QBAF. Its template based *argumentation dialogue* was first introduced in [3] and extended in [2]. The dialogue consists of *explanation requests* $\mathcal{Q}(\alpha)$ for $\alpha \in \mathcal{A}$ from the user, to which ADA responds with *explanations* $\mathcal{X}(\alpha)$. It is defined as follows, with use of the auxiliary functions in Figure 2.4:

$$\begin{aligned}
r_a^+(\alpha) &= \{\text{because (the) } \alpha \text{ was/were good}\}; \\
r_a^-(\alpha) &= \{\text{because (the) } \alpha \text{ was/were poor}\}; \\
r_b^+(\alpha) &= \{\text{although (the) } \alpha \text{ was/were good}\}; \\
r_b^-(\alpha) &= \{\text{although (the) } \alpha \text{ was/were poor}\};
\end{aligned}$$

$$r_a^+(\varnothing) = r_a^-(\varnothing) = r_b^+(\varnothing) = r_b^-(\varnothing) = \{\}.$$

Figure 2.4: Auxiliary functions for argumentation dialogue.

[2]https://www.rottentomatoes.com/m/star_wars_the_rise_of_skywalker

Definition 2.7 (Argumentation dialogue). For any $S \subseteq \mathcal{A}$, if $S = \varnothing$ let $max(S) = \varnothing$; else let $max(S) = argmax_{s \in S}\sigma(s)$. Then the dialogue is such that for any $\alpha \in \mathcal{A}$:

- if $\alpha = p$ and $\sigma(\alpha) > 0.5$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$Why was α highly rated?$\}$
  $\mathcal{X}(\alpha) = \{$This product was highly rated$\} + r_a^+(max(\mathcal{L}^+(\alpha))) + r_b^-(max(\mathcal{L}^-(\alpha)))$;

- else if $\alpha = p$ and $\sigma(\alpha) < 0.5$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$Why was α poorly rated?$\}$
  $\mathcal{X}(\alpha) = \{$This product was poorly rated$\} + r_a^-(max(\mathcal{L}^+(\alpha))) + r_b^+(max(\mathcal{L}^-(\alpha)))$;

- else if $\alpha \in \mathcal{F}$ and $\mathcal{V}^+(\alpha) > \mathcal{V}^-(\alpha)$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$Why was/were (the) α considered to be good?$\}$
  $\mathcal{X}(\alpha) = \{$(The) α was/were considered to be good$\} + r_a^+(max(\mathcal{L}^+(\alpha))) + r_b^-(max(\mathcal{L}^-(\alpha)))$;

- else if $\alpha \in \mathcal{F}$ and $\mathcal{V}^+(\alpha) < \mathcal{V}^-(\alpha)$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$Why was/were (the) α considered to be poor?$\}$
  $\mathcal{X}(\alpha) = \{$(The) α was/were considered to be poor$\} + r_a^-(max(\mathcal{L}^+(\alpha))) + r_b^+(max(\mathcal{L}^-(\alpha)))$;

- else if $\mathcal{V}^+(\alpha) > \mathcal{V}^-(\alpha)$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$What did users say about (the) α being good?$\}$
  $\mathcal{X}(\alpha) = \{[$phrase from $u \in \mathcal{U}$ constituting $\mathcal{V}(u, a) = +]\}$;

- else if $\mathcal{V}^+(\alpha) < \mathcal{V}^-(\alpha)$ and $\exists \beta \in \mathcal{L}^-(\alpha) \cup \mathcal{L}^+(\alpha)$ s.t. $\sigma(\beta) > 0$:

  $\mathcal{Q}(\alpha) = \{$What did users say about (the) α being poor?$\}$
  $\mathcal{X}(\alpha) = \{[$phrase from $u \in \mathcal{U}$ constituting $\mathcal{V}(u, a) = -]\}$.

When a user requests an explanation for the sentiment towards an argument α, ADA responds by providing another argument β that supports α, and possibly also contrasts it with a third argument γ that attacks α. A user can also request a direct quote from one of the reviewers on an argument. For example, a conversation between a user and ADA on the *Canon IXUS 185* digital camera[3] could take the following form:

User:	*Why was the Canon IXUS 185 highly rated?*
ADA:	*The product was highly rated because the lens was good, although the battery was poor.*
User:	*Why was the lens considered to be good?*
ADA:	*The lens was considered to be good because the autofocus was good, although the zoom was poor.*
User:	*What did users say about the zoom being poor?*
ADA:	*"...the zoom wasn't as good as advertised..."*

2.1.6 ADA for Amazon reviews

We will conclude this section by discussing the limitations of ADA in the Amazon review domain. To illustrate the differences between Amazon and Rotten Tomatoes reviews, let us take examples of a positive and a negative review from each site:

[3]https://www.amazon.co.uk/Canon-IXUS-185-Digital-Camera/dp/B01N6JP07Y

Amazon reviews

Canon IXUS 185 Digital Camera - Black
rated 4/5 stars

About the size of a box of 10 cigarettes. Really light fits into any trousers pocket good quality picture better than any of the cheaper models with the same ***specs****. I love* ***ot*** *because its a high enough quality to make it worth while taking it everywhere and* ***architecture being my interest*** *~I Take pictures of buildings all the time.*

Philips Sonicare Electric Toothbrush
rated 2/5 stars

So, I really enjoy how clean my teeth feel after using this toothbrush. It has a built in 2 minute timer which is also nice and the ***battery*** *stays charged for several days. My complaint is that when I changed the head of the toothbrush after the 3 month suggested timeframes, there was was mold in the handle of the base. I am considering purchasing something else because using something that contains mold close to my mouth isnt very appealing.*

Rotten Tomatoes reviews

Star Wars: The Rise of Skywalker (2019)
rated Fresh[4]

The pacing in the first act leaves something to be desired but at the end of the day, the joy I felt watching this beloved story come to its conclusion could not be diminished. The action, the ***characters****, and the overall world won me over one last time.*

A Rainy in New York (2019)
rated Rotten

Woody, too old to flirt with a pretty young thing, has split up the chore amongst multiple older ***characters****, as if that would somehow dilute the ick factor.*

The Amazon reviews are for the *Canon IXUS 185* digital camera and the *Philips Sonicare* electric toothbrush, and the Rotten Tomatoes reviews are for the movies *Star Wars: The Rise of Skywalker* (2019) and *A Rainy Day in New York* (2019). We notice some key differences between the Amazon Reviews and the Rotten Tomatoes reviews:

Features Because all Rotten Tomatoes reviews are on movies, there are features such as *characters*, found in both of the movie reviews, that are common to all movies. On the other hand, Amazon has reviews for a large variety of different products, and therefore the main features might be very different from one item to another. For example, the feature *cleanliness* is very important to an electric toothbrush but not to a digital camera, and vice versa for *image quality.* The vast amount of products limits the possibility of having predetermined features, and emphasises the importance of unsupervised feature extraction.

Writing style As the reviews on Rotten Tomatoes are written by critics, the style of writing tends to be similar across all reviews. On the other hand, Amazon reviews can be written by anyone who has purchased the product, so the style is more individualistic. For example in the review for the *Canon IXUS 185* digital camera, the user comments on their personal interests. Colloquial language (*'specs'*) and spelling mistakes (*'ot'*) are also common. Due to these differences, using the same off-the-shelf movie review classifier for sentiment analysis on Amazon reviews would not be sensible. Although ADA was explored in the user-written review domain of Trip Advisor in [2], this paper focused on expanding the conversational aspect of ADA with no consideration of its NLP methods.

Based on these two differences, we propose two extensions to ADA in order to accommodate Amazon reviews:

1. An unsupervised and generic method of feature extraction that will work for any product.
2. A method for sentiment analysis in Amazon's more heterogeneous review domain.

[4]Individual reviews on Rotten Tomatoes are rated Fresh (positive) or Rotten (negative).

2.2 Feature-level sentiment analysis

In this section, we will examine the state-of-the-art research in *aspect-level sentiment analysis* [27], which attempts to determine people's opinions on *entities* and their *aspects*. We have already been introduced to aspect-level sentiment analysis in ADA's review aggregation, where the entities are *products* and the aspects are their *features*. For consistency, we will refer to entities as products and aspects as features. Further research into this area will guide our implementation of the extensions proposed in Section 2.1.6.

Consider the following review for a particular model of the *Philips Sonicare* electric toothbrush range:

Trust me, as someone who has owned several terrible
Sonicare toothbrushes before: this model is great.

Since ADA would associate *Sonicare toothbrush* with *terrible*, it would incorrectly extract a negative vote for the product. This is because ADA:

1. cannot distinguish that the toothbrush under review is referred to as *this model*;

2. will calculate only a single sentiment for the entire phrase, while it actually represents a negative sentiment towards other Sonicare toothbrushes and a positive sentiment towards this particular toothbrush.

The first error has to do with *feature extraction*, which accounts for the first part of feature-level sentiment analysis where the agent extracts features from the text. In our case, we wish to extract also the relations between the various features, so we will concern ourselves with *ontology extraction*. The second error has to do with *feature-dependent sentiment analysis*, where the extracted features are assigned with sentiment polarities. The following sections evaluate research in these two areas in hopes of finding a way to resolve these errors.

2.2.1 Ontology extraction

In order to obtain an ontology for a product in an unsupervised manner, we must have a way to extract the features and their relations to one another from some source of information about the product, such as metadata or its review texts. As the availability of metadata on Amazon varies greatly from one product category to another, we shall focus our analysis on feature extraction from the review texts in order to not limit the set of Amazon products supported by our implementation.

Opinion target extraction

In order to construct an ontology for a product, we wish to extract the features of the product towards which the reviewers often have an opinion. In the field of *opinion target extraction*, various NLP methods are used to extract such words from natural language. Most studies on this topic can be categorised into *rule-based methods* or *supervised machine learning based methods* [5]. The former relies on lexical data or syntactic rules to detect features, such as in [22], while the latter models the problem as a sequence labelling task, often with the use of Conditional Random Fields (CRFs) [12]. More recently, deep learning methods have also been proposed for the problem, an overview of which can be found in [27].

When it comes to Amazon review texts, the issue with many of these methods is that they are domain-dependent. Particularly for the machine learning methods, the models work well on the domain on which they are trained on, but may face a performance drop of up to 40% when tested in different domains [5]. Therefore, a model trained on digital camera reviews might not perform well on reviews for garden hoses. On the other hand, it would be impossible to obtain manually labelled data that would be representative of the whole selection of millions of Amazon products.

Some cross-domain feature extraction methods have been proposed [5], but these still perform significantly worse than the single-domain methods discussed above. Furthermore, none of these methods look for semantic relationships between the product and its features, but search for opinion targets in the absence of any existing product information.

Common-sense knowledge bases

Semantic information can be obtained from *ConceptNet* [24], which is a *common-sense knowledge graph* connecting words and phrases with labelled and weighted edges expressing semantic relations between the words. As our goal is to obtain features of a product, we are most interested in the *HasA* relation. For example, the term *watch*[5] is related by the *HasA* relation to *a hand that counts minutes*. However, many reviews for watches comment on the watch's *face* or its *band*, neither of which are terms related to watches on ConceptNet. Furthermore, the knowledge on ConceptNet is loosely structured as natural language descriptions, whereas ADA requires more structured information: for a *watch*, we require the feature *hand* instead of *a hand that counts minutes*.

Some methods have been proposed for *automatic common-sense completion*, which aims to structure the information in common-sense knowledge graphs such as ConceptNet. *Common-sense Transformers* (COMeT) proposed by Bosselut et al. [1] is one such state-of-the-art method, which uses the Transformer architecture described in Section 2.3 to generate structured knowledge based on ConceptNet. However, even COMeT appears incomplete in terms of the *HasA* relation, as it is missing many of the same features of the term *watch* [6] as ConceptNet.

WordNet [19], a large manually constructed lexical database, is another popular source for common-sense lexical information. Although WordNet includes fewer relations than ConceptNet, it includes the relation of *meronymy*, which denotes a word being a constituent part of another word, similar to the *HasA* and *MadeOf* relations in ConceptNet. For example, the term *electric toothbrush*[7] has a single meronym of *electric motor*. As with ConceptNet, the list of relations on WordNet is incomplete, as it fails to include relevant meronyms such as the timer.

Due to the low recall of ConceptNet and WordNet in obtaining product features, our approach to ontology extraction must take advantage of the review texts as a source of information about the product. However, we can evaluate our implementation against ConceptNet and WordNet, or possibly take a hybrid approach to ontology extraction where features are mined from review texts to complement the existing semantic information on ConceptNet and WordNet.

2.2.2 Feature-dependent sentiment analysis

After we have extracted the features of a product, we wish to discern whether the opinions towards the features in reviews are positive or negative through sentiment analysis. Perhaps the main difficulty in feature-dependent sentiment analysis is to distinguish which opinions are acting on which arguments.

ADA attempts to tackle this issue by dividing the review into phrases at specific keywords, such as the word *but* in *I liked the acting, but the cinematography was dreadful*, after which it assumes each phrase contains at most one sentiment. However, there are many cases where such a simple method will not work, like the example at the start of this section. This is particularly true for Amazon reviews where the language tends to be less formal compared to Rotten Tomatoes reviews written by professional critics.

Various machine learning methods have been proposed for feature-dependent sentiment analysis, although the task is deemed difficult and there is currently no dominating technique for this purpose [27]. More traditional approaches, such as *SVM-dep* proposed by Jiang et al. [10], use hand-crafted syntactic features alongside word embeddings in *support-vector machine* (SVM) classifiers. More recently, various methods using deep learning have been proposed, such as *TD-BERT* by Gao et al. [6], which uses the BERT language model detailed in Section 2.3 to obtain state-of-the-art performance in the task.

However, both of these methods were trained and tested in the same domain including tweets or reviews about celebrities, companies and consumer electronics. The performance would likely drop substantially in a separate domain, as the sentiment polarity of a word can be highly dependent on context: for example the adjective *hard* has a positive connotation when describing a protective case, but a negative connotation when describing an armchair.

[5]http://conceptnet.io/c/en/watch
[6]https://mosaickg.apps.allenai.org/comet_conceptnet/?l=watch&r=HasA
[7]http://wordnetweb.princeton.edu

2.3 BERT

Both of the two NLP tasks relevant to ADA, feature extraction and feature-dependent sentiment analysis, become more difficult to perform in a general domain, which is crucial to applying ADA for all Amazon products. BERT [4], which stands for *Bidirectional Encoder Representations from Transformers*, is a state-of-the-art language representation model, which uses pre-training to learn language representations from large amounts of unlabelled text, which can then be fine-tuned for more specific NLP tasks. Because the unlabelled text used to pre-train BERT comes from a general domain corpus such as Wikipedia, the knowledge in a pre-trained BERT model is especially well-suited for domain-independent NLP tasks such as ours. This section will provide a brief overview of the BERT architecture and the underlying *Transformer network*, shown in Figure 2.5. The paper presents two model sizes for BERT, BERT-base and BERT-large, which differ in terms of the size of various hyperparameters. In this paper, we will use the former due to memory constraints.

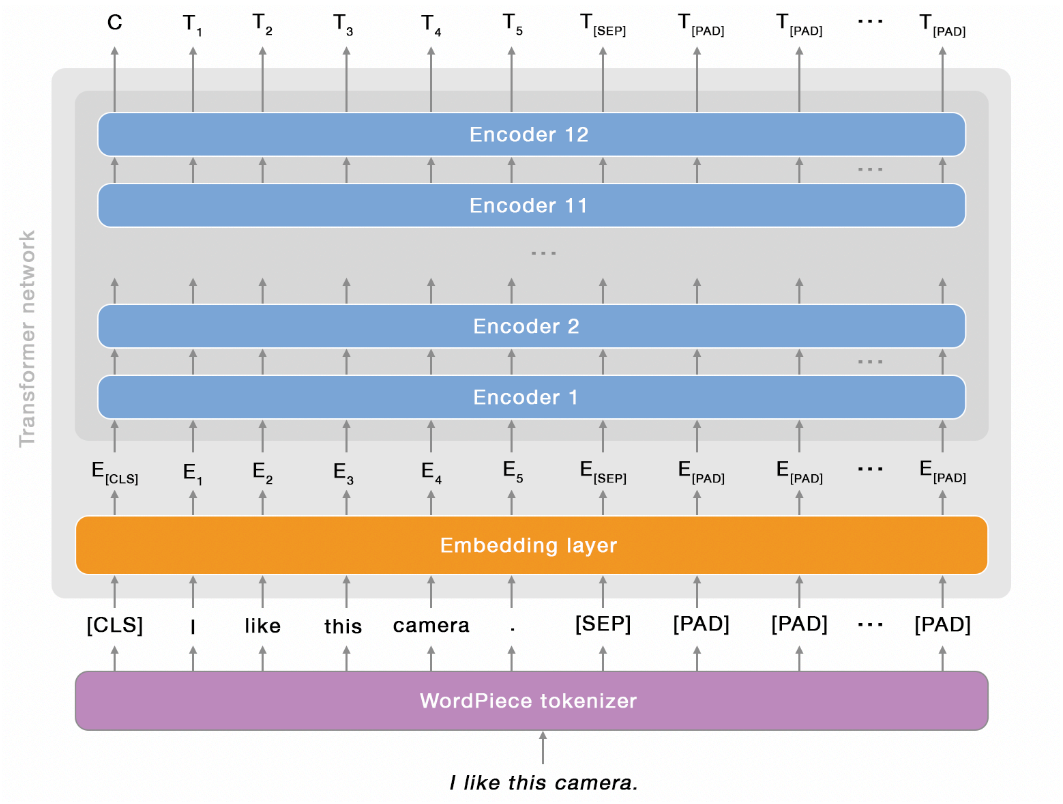

C
T_1 T_2 T_3 T_4 T_5 $T_{[SEP]}$ $T_{[PAD]}$ $T_{[PAD]}$ ··· $T_{[PAD]}$
Encoder 12
Encoder 11
···
Encoder 2
Encoder 1
Transformer network
$E_{[CLS]}$ E_1 E_2 E_3 E_4 E_5 $E_{[SEP]}$ $E_{[PAD]}$ $E_{[PAD]}$ ··· $E_{[PAD]}$
Embedding layer
[CLS] I like this camera . [SEP] [PAD] [PAD] ··· [PAD]
WordPiece tokenizer
I like this camera.

Figure 2.5: Overview of the BERT architecture

2.3.1 Tokeniser

The first step of BERT is to tokenise the input text in order to prepare it for the Transformer network, where each token follows its own path through the network. The tokenisation step in Figure 2.5 is a bit simplified, as BERT uses *WordPiece* tokenisation, which divides words into a limited set of common sub-word units, improving the handling of rare words. The first token is always the `[CLS]` token, which is used in the output as representative of the whole sequence of tokens. The `[SEP]` token is used to mark the end of the first sentence in tasks with two sentences, such as *Sentence Pair Classification*. Finally, the sequence of tokens is padded to a fixed length of up to 512 tokens, which enables BERT to handle batches including texts of varying sizes. The tokens are mapped to numerical IDs using a static mapping, after which the token IDs are passed onto the Embedding layer of the Transformer network.

2.3.2 Transformer network

The pre-trained network of BERT is based on an Embedding layer followed by a stack of Transformer encoders, proposed by Vaswani et. al. [25].

Embedding layer

Each token ID from the tokeniser is passed through an embedding layer, which outputs a vector $E_n \in \mathbb{R}^H$ of hidden size H (768 for BERT-base). The embedding is a sum of the *token embedding*, a *position embedding* for the token's position in the sequence, and a *segment embedding* which is used to distinguish between the first and second sentences in Sentence Pair Classification. The embeddings for each token are then passed onto the first encoder in the encoder network.

Encoder network

The Encoder network for BERT-base consists of a stack of 12 identical encoders. The input to the first encoder are the embeddings, and the output from the $(n-1)$th encoder is used as input to the nth encoder for $n \in [2, 12]$. Figure 2.6 shows a more detailed diagram of a single encoder in the encoder network. Each encoder consists of two parts, a *self-attention* layer and a feed-forward neural network.

Although the representations for each token follow their own path through the network, the self-attention layer is where they interact with each other in order to build a context-aware representation for each token. Let $X_{n-1} \in \mathbb{R}^{T,H}$ be a matrix containing the inputs to the encoder, where T is the number of tokens. Each self-attention layer learns three weight matrices, W_Q, W_K, and W_V used to calculate the Q (query), K (key), and V (value) matrices respectively:

$$\begin{aligned} Q &= X_{n-1} \times W_Q \\ K &= X_{n-1} \times W_K \\ V &= X_{n-1} \times W_V. \end{aligned}$$

Then, the outputs Z of the self-attention layer are calculated with the formula

$$Z = \texttt{softmax}(\frac{Q \times K^T}{\sqrt{d_k}}) \times V,$$

where d_k an arbitrary constant used to stabilise the gradient. The calculation $Q \times K^T$ returns for each token position (row of the matrix) the query vector of that position multiplied with the key vectors of all of the positions. The `softmax` operation then normalises the resulting rows so that each sums up to one; intuitively, the resulting rows represent how much each token position is expressed at the position of the row. Finally, we multiply the `softmax` probabilities for each position with the corresponding value vectors, in order to obtain the output Z. In essence, each row of Z is a weighted combination of all of the value vectors in V where the weights are determined by Q and K.

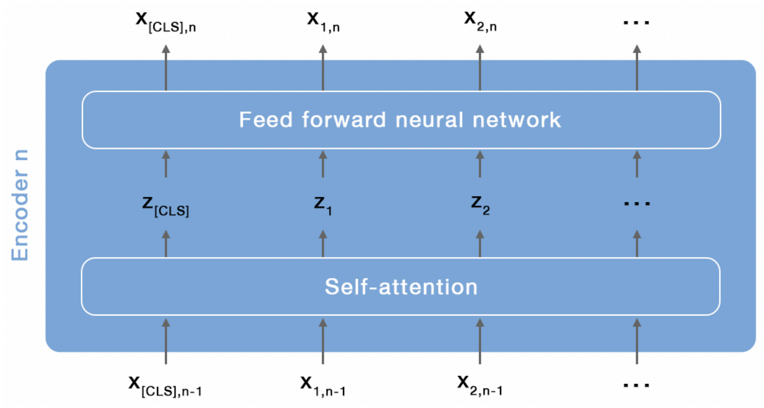


Figure 2.6: A single encoder

BERT implements *multi-headed* attention, which means that each self-attention layer has in fact 12 (16 for BERT-large) sets of query, key, and value matrices which are all randomly initialised and learn a different representation subspace. Therefore, we end up with 12 output matrices $Z_1, \ldots, Z_{12}$ which are concatenated and multiplied with an additional weight matrix W_O to obtain the final output matrix Z which captures information from all of the attention heads. Multi-headed attention improves the performance of the self-attention layer by allowing the model to focus on different positions in different representation subspaces.

The outputs of the self-attention layer are used as inputs to the feed-forward neural network, which returns outputs $X_n \in \mathbb{R}^{T,H}$ used as inputs to the next encoder. The same network is independently applied to each position, so the interaction between positions happens in the self-attention layer only.

2.3.3 Fine-tuning BERT

The outputs from the last encoder, which are denoted in Figure 2.5 as C for the `[CLS]` token position and T_n for the nth text token, are used as inputs to a classifier network depending on the specific NLP task. Many sentence-level classification tasks use simply the output C, which is considered representative of the whole sequence of tokens, while the outputs T_n are more representative of the nth token position.

While the weights for BERT can be loaded from pre-trained models, the classifier is trained from scratch for each task in what is called *fine-tuning* the model. Often, the weights in BERT are also fine-tuned alongside training the classifier, although the changes are minimal and BERT mostly relies on its pre-trained knowledge, which it passes onto the classifier.

2.4 Conversational systems

The main innovation of ADA lies in its ability to provide dialogical explanations for review aggregations, which help the user understand why a product is liked or disliked. This is a beneficial feature for example in the recommender system domain, where it has been shown to improve the general acceptance, perceived quality, and effectiveness of the system [7].

Although an argumentation dialogue has been defined for ADA, a conversational user interface through which a user can participate in this dialogue remains to be implemented. In this section, we will investigate two novel approaches to designing such an interface: *Botplications* proposed by Klopfenstein et al. [11] for textual interfaces and *Quantised Dialog* proposed by Gunasekara et al. [8] for speech interfaces. We will conclude by evaluating the two methods using the RRIMS properties proposed by Radlinski et al. [23] for conversational search systems.

2.4.1 Botplications

Klopfenstein et al. define a *Botplication* as "an agent that is endowed with a conversational interface accessible through a messaging platform, which provides access to data, services, or enables the user to perform a specific task". To extend ADA with such functionality, we would implement a messaging interface through which the user can request review explanations in line with the argumentation dialogue in Definition 2.7. Instead of demanding the user to type out a request, the interface might implement structured message forms, such as preset replies or an interactive list of available commands. The advantage of structured messages is that they constrain the conversation into a limited number of expected outcomes and assist the user in using the interface.

The alternative to structured messages would be to use NLP methods to extract commands and intent from the user's messages. However, Klopfenstein et al. argue that for a single bot, natural language should be avoided where possible, as "going after AI is mostly excessive and counterproductive, when the same results can be obtained with simple text commands using a limited structured language".

2.4.2 Quantised Dialogue

In a speech interface, structured messages are no longer possible. Although advances in deep learning have recently enabled speech recognition to reach human parity[8], understanding the intent behind the extracted natural language continues to be a challenging task and an active area of research. Many approaches to this problem in conversational systems are based on heuristics and handcrafted rules, which may not adapt well to unexpected input.

Gunasekara et al. recently proposed the more general method of *Quantised Dialogue* as a way to tackle this issue. It relies on the semantic quantisation and clustering of the user's requests in order to reduce the otherwise limitless interaction space of natural language dialog. It is then possible to teach a neural network in this simplified dialog space to predict the appropriate response to a query.

2.4.3 Evaluation using the RRIMS properties

Radlinski et al. propose that a conversational search system should satisfy the following five properties, termed the *RRIMS properties*:

- **User Revealment** The system helps the user express, and possibly discover, their true information need.
- **System Revealment** The system reveals to the user its capabilities and commands, building the user's expectations of what it can and cannot do.
- **Mixed Initiative** Both the system and the user can take initiative as appropriate.
- **Memory** The user can reference past statements, which implicitly also remain true unless contradicted.
- **Set Retrieval** The system can reason about the utility of sets of complementary items.

ADA's argumentation dialogue fairs well in terms of *user revealment*, as it allows the user to request additional information about the product and its features. ADA also provides additional information to contrast with the requested information via its secondary *although...* phrases, allowing the user to discover more about the product. For the user interface methods, Botplications perform better than Quantised Dialogue in terms of the *system revealment* and *memory* properties. In terms of *system revealment*, the structured messages of a Botplication provide more information about the system's capabilities. In terms of *memory*, the chronological log of past interactions kept by Botplications' messaging interface again seems more useful. We will not concern ourselves with the *mixed initiative* and *set retrieval* properties, as we will not be implementing ADA as a full-fledged conversational search system in this project.

As a Botplication evaluates better Quantised Dialogue in terms of fulfilling the RRIMS properties, the only reason we should implement the latter is if we specifically want a speech interface. Such an interface could be useful in for example a smart speaker, which could then provide the user with explainable product recommendations in situations where they are unable to interact with a physical device, such as while cooking. However, since the main focus of this project is implementing ADA in the general-domain, we will concentrate our efforts on developing a single Botplication interface.

[8] https://www.microsoft.com/en-us/research/blog/microsoft-researchers-achieve-new-conversational-speech-recognition-milestone/

Chapter 3

Ontology extraction

Since there does not exist a domain-independent common-sense ontology that would suit our purposes, we will extract the ontology ourselves from the review texts using NLP methods. In this paper, we will limit the extraction of features to unigram, bigram, and trigram nouns, as the vast majority of terms for products and features fall into these categories. Although not strictly necessary, this will greatly help limit our search for features within the review texts with little effect on the recall of our model. Furthermore, we limit the maximum depth of the ontology tree to two, as was the case in the paper of Cocarascu et al.

3.1 Implementation

Our method of ontology extraction is a multi-step pipeline using both hand-crafted grammatical features and two BERT-based models trained using *distantly supervised learning*. The only input to the system are review texts for the product for which we wish to extract an ontology. The first BERT model is used for *named-entity recognition* (NER) of the various features of the product, while the second model is used for *relation extraction* (RE) in order to extract sub-feature relations between the recognised features. In addition, we use a *Word2Vec* [17] model to extract word vectors, which are used to group the features into sets of synonyms, or *synsets*, using a method proposed by Leeuwenberg et al. [13]. The pipeline is structured as follows:

1. Noun extraction
2. Feature extraction
3. Synonym extraction
4. Ontology extraction.

In this section, we will first detail the annotation method used to obtain the training data for the two BERT-based models, after which we will go through the different pipeline steps in detail.

3.1.1 Annotation of training data for masked BERT

Annotating training data that would be representative of the whole set of Amazon products would be nearly impossible due to the sheer number of different product categories on Amazon. However, in review texts, certain grammatical constructs stay the same regardless of the product. Take for example the two review sentences:

I love the ***lens*** *on this* ***camera*** and *I love the* ***material*** *of this* ***sweater****.*

Clearly, *lens* is a feature of *camera* and *material* is a feature of *sweater*. If we mask the entities mentioned in the two sentences, we obtain:

I love the ***e1*** *on this* ***e2*** and *I love the* ***e1*** *of this* ***e2****,*

which are nearly identical. So while the entities in review texts are domain-specific, the context is often largely domain-independent. Therefore, using the masked sentences, it would be possible to train a classifier to recognise that *e1* and *e2* are both entities, or that *e1* is a sub-feature of *e2*.

In fact, BERT has inbuilt support for masking, as it plays a central role in its pre-training phase. One of the two tasks on which BERT is pre-trained is *masked language modelling*, in which randomly chosen words are replaced with a `[MASK]` token and the model is asked to predict the masked words in a large corpus of text.

Of course, the above example is an idealised scenario. If we mask the entities in the following review sentences:

The ***camera housing*** *is made of shiny black* ***plastic*** *but it feels nicely weighted and solid*
and
It's a lovely warm ***jumper*** *that has a nice* ***feel*** *to it,*

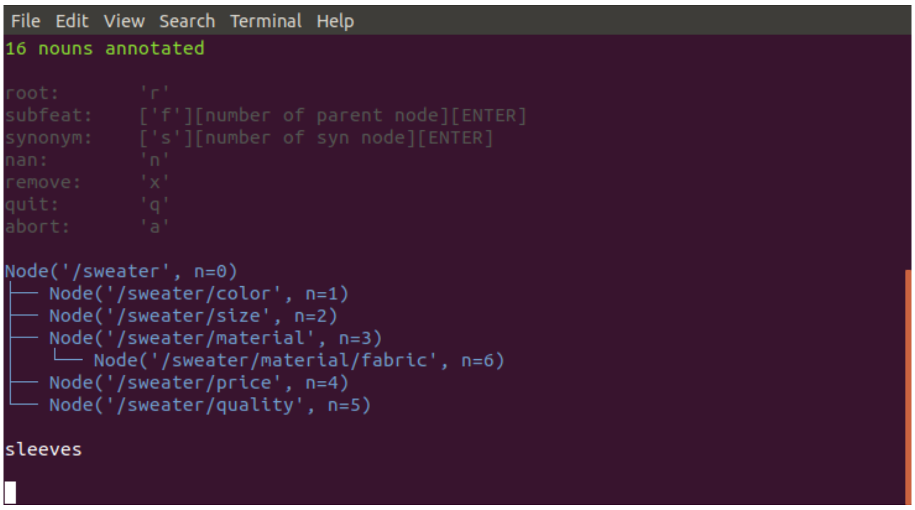
we obtain the masked sentences:

The ***e1*** *is made of shiny black* ***e2*** *but it feels nicely weighted and solid*
and
It's a lovely warm ***e1*** *that has a nice* ***e2*** *to it,*

which still contain some rather domain-specific terms such as *shiny*, *weighted*, and *warm*. However, these words are less likely to be product-specific, but are often common to wider categories of products, such as electronics or clothing. Therefore, there is no need to annotate each and every product to represent the whole set of Amazon products, but a small and varied set of products should be enough to train a domain-independent classifier.

Even annotating texts for just a few products would still require the annotation of at least thousands of texts in order to obtain a sufficiently large dataset. However, we can reduce the amount of work substantially by taking advantage of *distantly supervised learning*, where rather than annotating each and every sentence by hand, we automatically annotate a large amount of text using predetermined heuristics. In our case, the heuristic will be a manually labelled ontology for the product: using this ontology, an automated process can search review texts for terms that appear in the ontology and label as well as mask them correctly. This will allow us to easily annotate data for multiple products, as we will only have to annotate their ontologies, which usually consist of no more than a few dozen terms.

Notice that distant supervision is made possible here by the masking of the terms in the ontologies. We are annotating a much smaller set of terms than the size of the resulting training set, which means that the training set will be highly saturated with the annotated terms. Therefore, if the words weren't masked, the classifier would simply learn to recognise the terms in the ontology and completely ignore their context. However, due to the masking of the terms, the classifier is forced to rely solely on their context, which is more varied.



```
16 nouns annotated

root:       'r'
subfeat:    ['f'][number of parent node][ENTER]
synonym:    ['s'][number of syn node][ENTER]
nan:        'n'
remove:     'x'
quit:       'q'
abort:      'a'

Node('/sweater', n=0)
├── Node('/sweater/color', n=1)
├── Node('/sweater/size', n=2)
├── Node('/sweater/material', n=3)
│   └── Node('/sweater/material/fabric', n=6)
├── Node('/sweater/price', n=4)
└── Node('/sweater/quality', n=5)

sleeves
```

Figure 3.1: Entity annotator interface

A program was written to ease the annotation of the ontologies, the interface of which is shown in Figure 3.1, exemplified for sweaters. The program takes as input review sentences for a given product category such as sweaters, and counts the number of times each noun occurs in the sentences using the same method as the ontology extractor, which is detailed in Section 3.1.2. It then displays the nouns one by one to the annotator in descending order of count, and for each noun, the annotator will label it as either the root of the ontology tree (the product), a sub-feature or a synonym of an existing node in the tree, or *nan* for nouns that are not an argument (for example *daughter* in the review text *my daughter loves it*). In order to simplify the annotation process, a noun with a lower count can only be annotated as a sub-feature of a noun with a higher count, as the reverse is rarely true. The annotation of *nan* nouns is important as it allows us to obtain negative samples for the training data.

Once the annotator has labelled a sufficient number of nouns, the program can use the annotated ontology to label the entire set of review sentences to be used as training data for either the feature or relation extraction models.

For the feature extraction training data, the program will select sentences with exactly one of the annotated nouns and label the sentence with a binary label representing if the masked word is an argument or not. Although selecting only a subset of the sentences will limit the amount of training data, it allows us to reduce the NER task to a binary classification problem instead of a sequence-labelling problem. Furthermore, even relatively small product categories have large amounts of review data available, so we can afford to prune some of it in order to improve the precision of our model. Table 3.1 shows a positive and a negative example.

`text`	`noun`	`is_argument`
"I love this sweater."	"sweater"	1
"My daughter loves it!"	"daughter"	0

Table 3.1: Example training data for feature extraction

For the relation extraction training data, the program will select sentences with exactly two of the annotated nouns, and mask both of the nouns as well as label the sentence with one of three labels: 0 for no relation between the masked nouns, 1 if the second masked noun is a feature of the first masked noun, and 2 if the first masked noun is a feature of the second masked noun. A noun n_1 is considered a feature of noun n_2 iff n_1 is a descendant of n_2 in the annotated ontology tree. For example, *fabric* is considered a feature of both *sweater* and *material* based on the ontology tree in Figure 3.1. Table 3.2 shows examples for each of the labels.

`text`	`noun_1`	`noun_2`	`relation_label`
"I like the colour and the material."	"colour"	"material"	0
"My daughter loves this sweater."	"daughter"	"sweater"	0
"The sweater's fabric is so soft."	"sweater"	"fabric"	1
"The colour of the sweater is beautiful."	"colour"	"sweater"	2

Table 3.2: Example training data for relation extraction

We used the program to obtain training data for a variety of five randomly selected products: digital cameras, backpacks, laptops, guitars, and cardigans. The categorised review data was obtained from a public repository[1] by Jianmo et al. [20]. For each of these products, we annotate the 200 most common nouns, as we observe that most of the relevant features of the product will be included within this range. After resampling to balance out the number of instances for each of the classes, we obtained the training data shown in Table 3.3.

	Number of training instances	
	per product	total
Feature extraction	56,526	282,630
Relation extraction	25,110	125,550

Table 3.3: Training data counts for ontology extraction

[1]https://nijianmo.github.io/amazon/index.html

3.1.2 Noun extraction

The first step of our ontology extraction method is to extract the most commonly appearing nouns in the review texts, which will be candidates for features in the following step.

The review data is divided into review texts, many of which are multiple sentences long, so we first split the texts into sentences. In this paper, we will treat each sentence as an individual unit of information, independent from other sentences in the same review text. We will then tokenise the sentences, and use an out-of-the-box implementation of a method by Mikolov et al. [18] to join common co-occurrences of tokens into bigrams and trigrams. This step is crucial in order to detect multi-word nouns such as *operating system*, which is an important feature of *computer*. After this, we use a part-of-speech tagger to select the nouns within the tokens, and count the number of occurrences for each of the nouns. Finally, as for the annotation method detailed in Section 3.1.1, we select the 200 most common nouns and pass them onto the feature extraction step.

3.1.3 Feature extraction

For the feature extraction step, we obtain review sentences that mention exactly one of the nouns obtained in the previous step, and pass the sentences through a BERT-based classifier to obtain votes for whether the noun is an argument or not. In the end, we aggregate these votes for each of the nouns and filter a list of extracted arguments.

BERT for feature extraction

Figure 3.2 shows the architecture of the BERT-based classifier used for feature extraction. The classifier takes as input a review sentence, as well as the noun which we wish to classify as an argument or a non-argument. The tokenisation step masks the tokens associated with the noun ('operating' and 'system') with the `[MASK]` token. The tokens are passed through the transformer network, and the output used for classification is taken from the positions of the masked tokens. The input to the linear classification layer is always of the dimension of a single BERT hidden layer output, so if there are several masked tokens, a max-pooling operation is performed on their outputs. The linear layer is followed by a softmax operation, which outputs the probabilities p_0 and p_1 of the masked noun being a non-argument or an argument, respectively.

For each of the nouns, we take the mean of its p_1 votes, and accept it as an argument if the mean is above 0.65, a hyperparameter tuned through validation to strike a good balance between precision and recall of the feature extraction. Using the raw output probabilities from the network rather than binary votes allows us to bias the aggregate towards more certain predictions of the model.

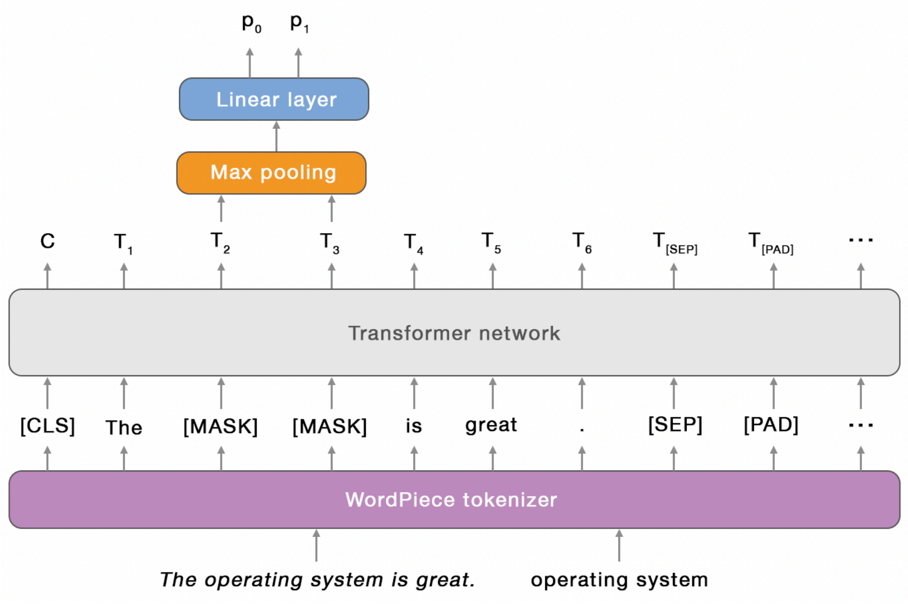


Figure 3.2: BERT for feature extraction

Training the model

We trained the model on the feature extraction data shown in Table 3.3, setting aside 5% of the data for validation. The final model was trained for 3 epochs with a batch size of 32. We used the Adam optimiser with standard cross entropy loss. The model was trained on a NVIDIA GeForce GTX 1080 GPU with 16GB RAM and took 3 hours and 16 minutes. The final accuracy and macro F1-score on the validation set were 89.70 and 89.39, respectively.

3.1.4 Synonym extraction

Reviewers can refer to the same argument using many different terms; for example the argument *laptop* can be referred to with the terms *computer*, *device*, and *product*. In order to construct an ontology tree, we must be able to group all of these terms under the same node. However, terms like *laptop* and *product* are not synonyms in the strict sense of the word, even if they are interchangeable within the review texts. Therefore, we cannot use a pre-existing synonym dictionary to group the arguments.

However, since the terms are interchangeable within the review texts, we can once again utilise the context of the words to group words with similar contexts into synsets. In order to compare the contexts of words, we must obtain context-based representations for them. One such representation is called a *word embedding*, which is a high-dimensional vector in a vector space where similar words are close to each other. We can obtain review-domain word embeddings by training a *Word2Vec* model on the review texts. The Word2Vec model learns the word embeddings by attempting to predict each word in the text corpus from a window of surrounding words.

We use a relatively small window of 4 words, exemplified by the following two review sentences where the window is underlined for the terms *laptop* and *product*:

I would recommend this ***laptop*** *to my friends, although the keyboard isn't perfect*
and
I would recommend this ***product*** *to my friends, as it is the best purchase I've ever made.*

The windows for *laptop* and *product* are identical, which means that their word embeddings will be similar. The small window ensures that the focus is on the interchangeability of the words, rather than their relatedness on larger scale. As the above two sentences illustrate, the terms *laptop* and *product* might be used in slightly different contexts on a larger scale, but their meaning, which is expressed in the nearby text, stays the same. Furthermore, the small window size prevents sibling arguments from being grouped together based on their association with their parent argument, as exemplified in these two review texts:

I like this lens because of the convenient ***zoom*** *functionality which works like a dream*
and
I like this lens because the quality of its ***glass*** *takes such clear pictures.*

Although both *zoom* and *glass* are mentioned in association with their parent argument *lens*, their nearby contexts are very different.

Once we have obtained the word embeddings, we can use the *relative cosine similarity* of the vectors to group them into synsets, as proposed by Leeuwenberg et al. in [13], who showed that relative cosine similarity is more accurate of a measure for synonymy than cosine similarity. The cosine similarity relative to the top n most similar words between word embeddings w_i and w_j is calculated with the following formula:

$$rcs_n(w_i, w_j) = \frac{cosine\_similarity(w_i, w_j)}{\sum_{w_c \in TOP_n} cosine\_similarity(w_i, w_c)},$$

where TOP_n is a set of the n most similar words to w_i. In this paper, we use $n = 10$. If $rcs_{10}(w_i, w_j) > 0.10$, w_i is more similar to w_j than an arbitrary similar word from TOP_{10}, which was shown in [13] to be a good indicator of synonymy.

Let arguments a_1 and a_2 be synonyms if $rcs_{10}(a_1, a_2) + rcs_{10}(a_2, a_1) \geq 0.21$. Then we group the arguments $\mathcal{A}$ into synsets $\mathcal{S}$ where

$$\forall a_1, a_2 \in \mathcal{A}.\ \forall s \in \mathcal{S}.\ rcs_{10}(a_1, a_2) + rcs_{10}(a_2, a_1) \geq 0.21 \land a_1 \in s \implies a_2 \in s,$$

and

$$\forall a \in \mathcal{A}.\ \exists s \in \mathcal{S}.\ a \in s.$$

3.1.5 Ontology extraction

The synsets obtained in the previous step will form the nodes of the ontology tree. In this step, we will extract the sub-feature relations that will allow us to construct the shape of the tree. In order to do this, we obtain review sentences that mention a word from exactly two synsets, and pass the sentences through a BERT-based classifier to obtain votes for whether the arguments are related, and if they are, which of the arguments is a feature of the other. In the end, we aggregate these votes within each of the synsets to obtain a relatedness measure between each of the synset pairs, which we use to construct the ontology.

BERT for relation extraction

Figure 3.3 shows the architecture of the BERT-based classifier we use for relation extraction. The classifier takes as input a review sentence, as well as the two arguments a_1 and a_2 for which we wish to obtain one of three labels: 0 if a_1 and a_2 are not related, 1 if a_2 is a feature of a_1, and 2 if a_1 is a feature of a_2. The tokenisation step masks the tokens associated with the arguments ('laptop', 'operating', and 'system') with the `[MASK]` token. The tokens are passed through the transformer network, and the output used for classification is taken from the positions of the masked tokens for the two arguments. If an argument consists of several tokens, a max pooling operation is performed on the outputs for each of the tokens such that we obtain a single vector for both arguments. The vectors are then concatenated and passed onto a linear classification layer with an output for each of the three labels. The linear layer is followed by a softmax operation, which outputs the probabilities p_0, p_1, and p_2 of the three labels.

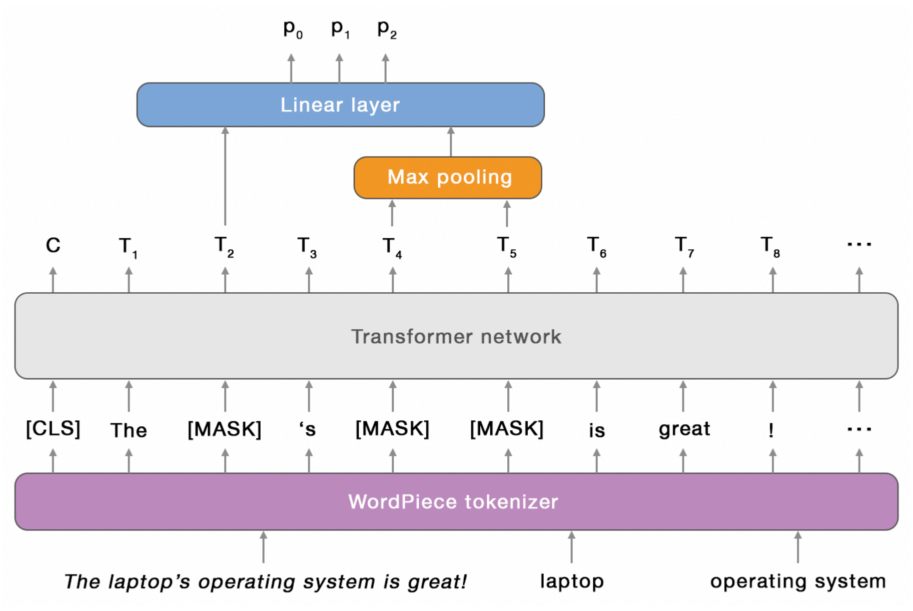

p0 p1 p2
Linear layer
Max pooling
C T1 T2 T3 T4 T5 T6 T7 T8 ···
Transformer network
[CLS] The [MASK] 's [MASK] [MASK] is great ! ···
WordPiece tokenizer
The laptop's operating system is great!
laptop
operating system

Figure 3.3: BERT for relation extraction

Training the model

We trained the model on the relation extraction data shown in Table 3.3, setting aside 5% of the data for validation. The final model was trained for 3 epochs with a batch size of 16. We used the Adam optimiser with standard cross entropy loss. The model was trained on a NVIDIA GeForce GTX 1080 GPU with 16GB RAM and took 2 hours and 5 minutes. The final accuracy and macro F1-score on the validation set were 83.40 and 82.03, respectively.

Ontology construction from votes

Let N be the number of synsets and $V \in \mathbb{N}^{N \times N}$ be a matrix where we accumulate the relation votes between each of the synsets. V is initialised with zeroes, and for each vote (p_0, p_1, p_2) on arguments belonging to synsets s_n and s_m we accumulate the element $v_{m,n}$ of V by p_1 and the element $v_{n,m}$ by p_2. In the end, element $v_{i,j}$ of V contains the sum of votes for s_i being a feature of s_j.

Let $n_{i,j}$ be the total number of input sentences to the relation classifier with arguments from s_i and s_j. Then

$$\bar{v}_{i,j} = \frac{v_{i,j}}{n_{i,j}}$$

is the mean vote for s_i being a feature of s_j. However, this is not a reliable measure of relatedness on its own, as many unrelated arguments might only appear in a few sentences together, which is not enough data to guarantee an accurate representation of their relatedness. On the contrary, if a_1 is a feature of a_2, a_1 is likely to appear often in conjunction with a_2. We can use this observation to improve the accuracy of the relatedness measure.

Let c_i be the total count of occurrences of an argument from s_i in the review texts. Then

$$\tau_{i,j} = \frac{n_{i,j}}{c_i}$$

is a relative measure for how often an argument from s_i appears in conjunction with an argument from s_j. If we scale $\hat{v}_{i,j}$ by $\tau_{i,j}$, we obtain a more accurate measure for relatedness,

$$r_{i,j} = \hat{v}_{i,j} \times \tau_{i,j} = \frac{v_{i,j}}{c_i}.$$

Using this formula, we define the *relation matrix*

$$R = V \,/\, \mathbf{c},$$

where $\mathbf{c}$ is a vector containing the counts c_i for each $s_i \in S$.

We know that the product itself forms the root of the ontology tree, so we do not have to consider the product synset being a sub-feature of another synset. For each of the remaining synsets s_i, we calculate its super-feature $\hat{s}_i$ using row r_i of the relation matrix, which contains the relatedness scores from s_i to the other synsets. For example, the row corresponding to the synset of *numbers* for the product *watch* could be as follows:

watch	dial	band	battery	numbers	quality
0.120	0.144	0.021	0.041	-	0.037

Clearly, *numbers* appears to be a feature of *dial*, as the relatedness score for *dial* is higher than for any other feature. Also the relatedness score for the product *watch* is high, as is expected for any feature since any descendant of a product in the ontology is considered its sub-feature, as defined in Section 3.1.1. Based on experimentation, we define $\hat{s_i} = s_j$ where $j = argmax(r_i)$, although other heuristics could work here as well.

Using the super-feature relations, we build the ontology tree from the root down with the function shown in pseudocode in Figure 3.4.

```
def get_tree(R, synsets):
    root = synsets.pop(product)  # set product synset as root

    # insert all direct children of product
    for s in synsets if s.super == product:
        add_child(root, synsets.pop(s))

    for s in synsets sorted in descending order by R[s][s.super]:
        if descendant(tree, s.super):
            # super-feature of s already in tree
            if depth(s.super) < 2:
                add_child(s.super, s)
            else:
                # max depth would be exceeded so set as sibling instead
                add_child(parent(s.super), s)
        else:
            # super-feature of s not yet in tree
            add_child(root, synsets.pop(s.super))
            add_child(s.super, s)

    return root
```

Figure 3.4: Function for constructing the ontology tree

3.2 Evaluation

In this section, we evaluate our ontology extraction method against ontologies extracted using WordNet and COMeT using human annotation. Furthermore, we evaluate the generality of the masked BERT models by experimenting with the number of the product categories used for their training.

3.2.1 Ontology evaluation

We evaluate five ontologies extracted for a variety of randomly selected products which were not included in the training data for the classifier: *watches*, *televisions*, *necklaces*, *stand mixers*, and *video games*. For each product, we use 200,000 review texts as input to the ontology extractor, except for *stand mixer*, for which we could only obtain 28,768 review texts due to it being a more niche category.

We also extract ontologies for the five products from WordNet[2] and COMeT[3] for comparison. For WordNet, we build the ontology top-down starting from the product term. A term t_f is considered a feature of t_p, if t_f is a meronym of either t_p or one of its direct *hyponyms* (specialisations of t_p, for example *camera* and *digital camera*). Using the web interface for COMeT, we are only able to obtain five directly related terms for the product with the *HasA* relation, which means that the ontologies for COMeT will not include any second-level sub-features of features (mining sub-features of features independently of the product context would lead to inaccurate results, as for example *face* has other meanings apart from its meaning in the context of the product *watch*).

The number of relations in each extracted ontology are shown in Table 3.4, while the full ontologies are included in Appendix A.

	watch	television	necklace	stand mixer	video game	total
Our method	26	22	20	17	7	92
WordNet	6	7	1	6	0	20
COMeT	5	5	5	5	5	25

Table 3.4: Number of extracted relations for the three ontology extraction methods

Precision

Since it is difficult to define a "complete" ontology for a product, we concentrate our measurement on the precision of the extracted ontologies. We will measure the precision of an ontology by the aggregated precision of its individual relations, which we will obtain by human annotation.

We present each of the 137 *has feature* relations in the ontologies to 3 human annotators, and ask them to annotate the relation as either true of false in the context of Amazon products. The context is important, as features such as *price* might not otherwise be considered a feature of a product. All three annotators agreed for 70.8% of the relations. Using the majority vote among the annotators, we calculate the precision for each of the three methods and five products, and present the results in Table 3.5 along with the total precision calculated for all 137 relations.

	watch	television	necklace	stand mixer	video game	total
Our method	88.46	86.36	70.00	88.24	100.00	84.78
WordNet	100.00	100.00	100.00	83.33	-	95.00
COMeT	60.00	40.00	60.00	40.00	20.00	44.00

Table 3.5: Precision scores for the three ontology extraction methods

Our method achieves a total precision of 84.78, which is comparable to the in-domain validation accuracies of the entity and relation extractors (89.70 and 83.40, respectively). We note that the precision for the stand mixer ontology is equivalent to the rest of the ontologies despite using less data, which suggests that our method is effective even for products with relatively little review data.

[2]http://wordnetweb.princeton.edu/perl/webwn

[3]https://mosaickg.apps.allenai.org/comet_conceptnet

WordNet obtains the highest total precision score of 95.00, which is expected since its knowledge has been manually annotated by human annotators. However, WordNet extracted on average only 4 relations for each ontology, while our method extracted on average 18.4 relations. Part of this could be due its outdatedness, as its last release was nine years ago in June 2011[4], although many of the products included in the comparison are quite timeless (*necklace*, *watch*). Furthermore, we observe that many of the terms extracted from WordNet, although correct, are scientific rather than common-sense (*electron gun*, *field magnet*), and therefore unsuitable for use in the Amazon review context.

The precision of our method is almost twice as good as the precision of the top five terms extracted by COMeT. Most of the erroneous relations for COMeT are either remnants of the unstructured information on ConceptNet (*game–effect of make you laugh*), or incorrectly categorised relations (*watch–hand and wrist*). Note that the precision of COMeT should not be affected by its limitation to direct features of the product.

Relative recall and F1-score

Let $R_{p,m}$ be a set containing the relations labelled as true by the majority in the ontology extracted by method $m \in$ [Ours, WordNet, ConceptNet] for product $p \in$ [*watches*, *televisions*, *necklaces*, *stand mixers*, and *video games*]. Assume the true ontology O_p for a product p is defined as follows:

$$\forall p, m.\ (a_1, a_2) \in R_{p,m} \iff (a_1, a_2) \in O_p.$$

Using this assumption, we calculate the *relative recall* $A_{m,p}$ for a method m and product p as:

$$A_{m,p} = \frac{|R_{p,m}|}{|O_p|}.$$

Although O_p might not represent the entire ontology for the product, we propose that $A_{m,p}$ is a good measure of relative recall between each of the methods, as any additions to O_p would decrease $A_{m,p}$ for all m. We calculate the relative recall for each of the three methods and five products, and present the results in Table 3.6 along with the total relative recalls. As we did not extract any sub-features for COMeT, we will only include direct features of the products in the O_p used for its recall calculations.

The total relative recall of our method is over four times greater than those of the other two methods. This suggests that while WordNet has a slightly higher precision than our method, it misses many of the key features of the products.

	watch	television	necklace	stand mixer	video game	total
Our method	88.46	79.17	77.78	71.43	87.50	80.41
WordNet	19.23	20.83	5.56	23.81	0.00	19.59
COMeT	17.65	11.11	23.08	16.67	14.29	16.42

Table 3.6: Relative recall scores for the three ontology extraction methods

Finally, to measure the overall performance of each method, we use the precision and relative recall scores to calculate F1 scores for each method. The results are shown in Table 3.7 alongside the macro-averaged F1 score across all five categories of products. We notice that the macro-F1 score of COMeT (23.64) actually surpasses that of WordNet (22.86), due to WordNet's poor recall in the *necklace* and *video game* categories. Our method is clearly superior than the other two methods, as its macro-F1 score (83.41) is nearly four times as high as those of the the other methods.

	watch	television	necklace	stand mixer	video game	macro-F1
Our method	88.46	82.62	73.69	78.95	93.33	83.41
WordNet	32.26	34.48	10.53	37.04	0.00	22.86
COMeT	27.28	17.39	33.34	23.53	16.67	23.64

Table 3.7: F1 scores for the three ontology extraction methods

[4]https://wordnet.princeton.edu/news-0

Annotation reliability

In order to assess the reliability of agreement between the annotators, we calculate the *Fleiss' kappa* measure κ, which calculates the degree of agreement over the degree expected by chance. The value of κ ranges from -1 to 1, with value 1 signalling total agreement and -1 total disagreement. The kappa measure is generally thought of being a more reliable measure of inter-rater reliability than simple percent agreement, as it takes into account the probability of agreement by chance. We obtain $\kappa = 0.417$, which in a well-known study of the coefficient [16] was interpreted to signify a weak level of agreement. This suggests that accurately determining *feature of*-relations is difficult even for humans, which validates the high precision score obtained by our method.

3.2.2 Generalisation evaluation

In this section, we evaluate the ability of our masked BERT method to generalise for the whole domain of Amazon products. In order to do this, we train the entity and relation classifiers with five incremental datasets $t_1 \dots t_5$, where t_n includes reviews from n different categories of products. The products included in each dataset are shown in Table 3.8. We evaluate the models using an unseen dataset w_e, which we have labelled for a sixth domain (watches). In addition, we train entity and relation classifiers on a separate watch dataset w_t, which can be evaluated with w_e to obtain an in-domain score. Each of the datasets contains a total of 50,000 instances, and all models were trained with the hyperparameter values used in Sections 3.1.3 and 3.1.5.

Dataset	Products included
t_1	cameras
t_2	cameras, backpacks
t_3	cameras, backpacks, laptops
t_4	cameras, backpacks, laptops, acoustic guitars
t_5	cameras, backpacks, laptops, acoustic guitars, cardigans
w_e	watches
w_t	watches

Table 3.8: Products included in each of the datasets

The evaluation accuracies for each of the classifiers trained on the five datasets $t_1 \dots t_5$ are plotted in Figure 3.5. The in-domain accuracies obtained by the classifiers trained using the dataset w_t are plotted as dashed lines. The accuracies for both entity and relation extraction increase significantly when trained with reviews for two products instead of just one, after which the accuracies appear to stay somewhat constant around 5 percentage units below the in-domain accuracies. The initial increase of accuracy with number of training products is expected, since a product-specific dataset will encourage the classifier to learn product-specific features. However, it is surprising to note that training the classifier with just two products (*camera* and *backpack*) is enough to raise its accuracy to its domain-independent optimum.

It appears that the domain-specific classifier has an advantage of around 5 percentage units over the domain-independent classifier. This can be attributed to various domain-specific context features the classifier can learn to take advantage of, such as domain-specific adjectives like *swiss* or *waterproof* for *watch*.[5]

[5]It is interesting to note that the domain-independent optimum lies approximately halfway between the initial accuracy and the domain-specific accuracy. When the classifier is trained on several products, it 'forgets' its domain-specific knowledge, which results in worse accuracy in its own domain but better accuracy in the unseen domain, as its knowledge becomes more general. It makes intuitive sense that the point of context-independence lies in between the two context-specific opposites.

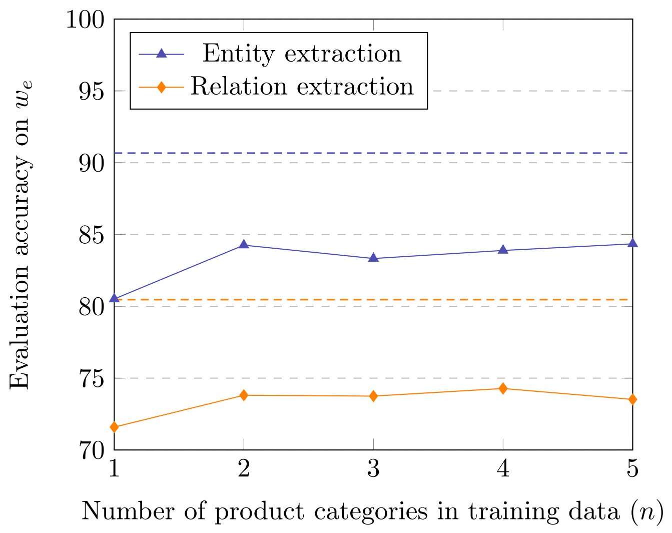


Figure 3.5: Accuracies on the w_e test set for masked BERT models trained with datasets $t_1 \dots t_5$ containing n categories of products. The accuracies of the models trained with in-domain data w_t are shown as dashed lines.

Chapter 4

Feature-dependent sentiment analysis

Unlike for our ontology extraction task, there already exists sufficiently accurate methods for our feature-dependent sentiment analysis purposes. In this paper, we have chosen to implement the state-of-the-art method of *TD-BERT* proposed by Gao et al. [6], trained on the *SemEval-2014 Task 4* [21] data for *Aspect Based Sentiment Analysis*. Since the data is domain-specific, we experiment with modifications to TD-BERT in order to improve its general domain performance.

4.1 Availability of data

As opposed to its sentence-level counterpart, feature-dependent sentiment analysis is a word-level task, which means that obtaining data for it is highly difficult. In practise, the sentiment towards each entity mention in the data has to be manually labelled by several human annotators, as the correct label is not always evident. As training a neural network requires extensive amounts of training data, we do not have the resources to annotate this data ourselves, but have to rely on an external source of data.

The most well-known available data for the task comes from Task 4 of the SemEval-2014 evaluation series, which was based around feature-dependent sentiment analysis. The data[1] consists of manually annotated customer review sentences for laptops and restaurants, with the training and testing data counts shown in Table 4.1. An example from the laptop dataset is shown in Table 4.2.

Dataset	Number of sentences	
	training	testing
laptops	3045	800
restaurants	3041	800
combined	6086	1600

Table 4.1: SemEval-2014 Task 4 data counts

`sentence`	`term`	`polarity`	`from`	`to`
"I charge it at night and skip taking the cord with me because of the good battery life."	"cord"	"neutral"	41	45
	"battery life"	"positive"	74	86

Table 4.2: Example of SemEval-2014 Task 4 data

Each aspect term in the data is labelled with one of four labels according to the sentiment expressed towards it: *neutral*, *positive*, *negative*, or *conflict*. The *conflict* label is given to terms such as *screen* in *"small screen somewhat limiting but great for travel"*, where both positive and

[1] Available at http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

negative sentiments are expressed. Only a small proportion of the training data is annotated with the *conflict* label (45 for laptops), so it will be difficult for the classifier to learn to accurately predict this category.

This dataset suits our purposes as it is in the domain of user reviews, particularly the dataset for laptops, since they are consumer products that can be found on Amazon. However, the dataset is quite small for training a neural network, so we include also the restaurant dataset in our evaluation. Furthermore, our experiments in Section 3.2.2 showed that training a classifier on just two domains can significantly improve its performance in the general domain.

4.2 Implementation

In this section, we will detail our implementation of the TD-BERT architecture. In addition, we will propose an extension similar to the masking method used in Sections 3.1.3 and 3.1.5 in hopes of improving the general domain performance of TD-BERT.

4.2.1 TD-BERT

The architecture of TD-BERT, shown in Figure 4.1, is similar to the two BERT architectures we have seen so far, in that it max pools the output at the positions of the tokens we are interested in. In this case, we select the output at the position(s) of the feature for which we wish to obtain a sentiment. The pooling is followed by a linear classification layer and a softmax operator, which outputs the the probabilities for the four possible cases: p_{neut} for neutral sentiment, p_{pos} for positive sentiment, p_{neg} for negative sentiment, and p_{conf} for conflicted sentiment.

Masked TD-BERT

Because we will be training the model with data for only laptop and restaurant reviews, we expect for it to be biased towards those domains. In attempt to alleviate this bias, we propose the addition of entity masking to TD-BERT. As for the BERT models used for ontology construction, we replace the tokens for the entity with `[MASK]` tokens. For example, in Figure 4.1, tokens *battery* and *life* would be masked.

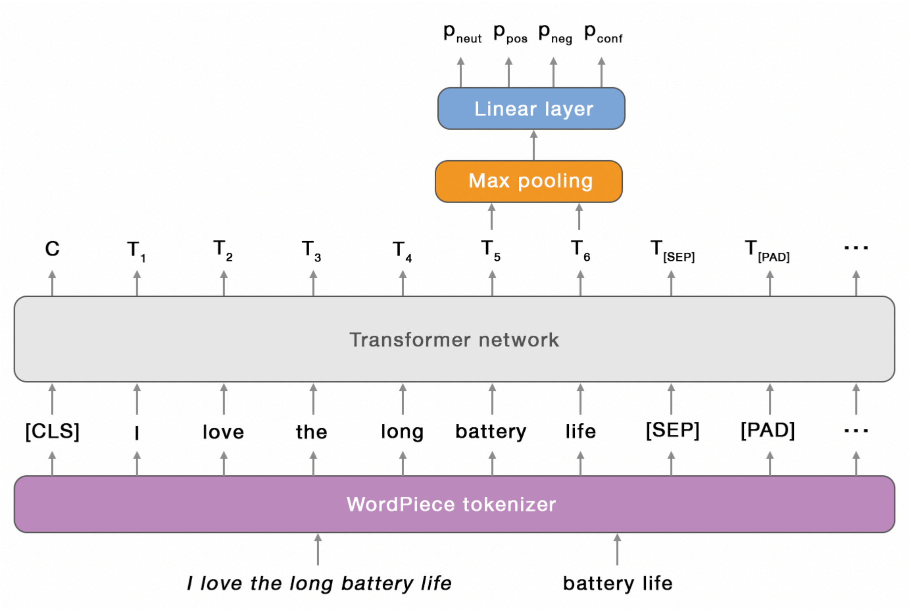

p_neut p_pos p_neg p_conf
Linear layer
Max pooling
C T_1 T_2 T_3 T_4 T_5 T_6 T_[SEP] T_[PAD] ...
Transformer network
[CLS] I love the long battery life [SEP] [PAD] ...
WordPiece tokenizer
I love the long battery life
battery life

Figure 4.1: TD-BERT architecture

Training the models

We trained a total of four different models on the SemEval training data shown in Table 4.1. We trained two models each for both unmasked and masked TD-BERT, one with training data for just laptops, and one with a combined training dataset for both laptops and restaurants.

We used the same hyperparameters as Gao et al.: each model was trained for 6 epochs with a batch size of 32, using the Adam optimiser with standard cross entropy loss. The models were trained on a NVIDIA GeForce GTX 1080 GPU with 16GB RAM.

4.3 Evaluation

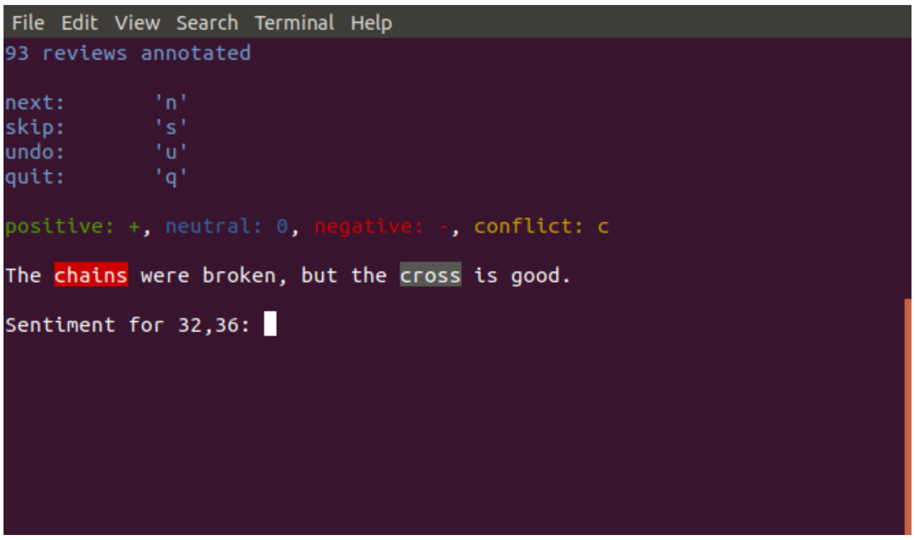
We evaluate the models using the in-domain SemEval-2014 testing data shown in Table 4.1, as well as a general domain review dataset we annotate ourselves for a selection of Amazon product reviews.

4.3.1 Annotation of Amazon sentiment analysis data

We write a simple program, shown in Figure 4.2, to help us annotate Amazon review data for the evaluation of the sentiment analysis models. Using this program we annotate a set of 100 Amazon reviews consisting of 20 reviews from five product categories: *watches*, *televisions*, *necklaces*, *stand mixers*, and *video games*. We annotate a total of 285 entities across 481 sentences, obtaining the data shown in Table 4.3.

Sentiment	Number of entity instances
neutral	71
positive	133
negative	65
conflict	16

Table 4.3: Amazon sentiment analysis evaluation data counts



```
93 reviews annotated

next:       'n'
skip:       's'
undo:       'u'
quit:       'q'

positive: +, neutral: 0, negative: -, conflict: c

The chains were broken, but the cross is good.

Sentiment for 32,36: 
```

Figure 4.2: Entity sentiment annotator interface

4.3.2 Results

We evaluate each of the four models on both the SemEval-2014 testing data as well as the Amazon data, and display the accuracies and macro-F1 scores in Table 4.4. For the SemEval-2014 testing data, we use the laptop testing data to evaluate the models trained on the laptop training data, and similarly for the combined data, in order to evaluate the in-domain performance of the models.

We note that the macro F-1 scores are significantly lower than the accuracy scores for all cases. This is due to the little training data available for the *conflict* category: none of the models predicted any *conflict* labels so the recall for the category was zero.

Method	Training data	SemEval-2014		Amazon	
		Accuracy	Macro-F1	Accuracy	Macro-F1
Unmasked	laptops	77.22	56.44	74.65	52.85
	combined	79.92	55.84	78.52	58.41
Masked	laptops	75.54	54.79	77.82	58.26
	combined	79.92	56.20	75.00	55.54

Table 4.4: TD-BERT sentiment analysis evaluation results

In-domain evaluation with SemEval-2016 data

We note that for both models, performance is significantly higher when trained and tested on the combined set of laptops and restaurants as opposed to just laptops. This is likely due to restaurant reviews being an easier domain for sentiment classification. As expected, the unmasked method performs better than the masked method when trained and evaluated in-domain with laptops. For the combined case, the two methods performed equally well in terms of accuracy, while the masked method obtained a slightly higher Macro-F1 score. This echoes the claim from Section 3.2.2 that the in-domain advantage of the unmasked method is lost even in a domain of just two different products.

Out-of-domain evaluation with Amazon data

The model trained on the combined data with the unmasked method appears to perform best in the out-of-domain evaluation, achieving an accuracy of 78.52, which is not far behind its state-of-the-art accuracy of 79.92 for the in-domain evaluation. The accuracy is significantly higher than that of the unmasked model trained on the laptop data alone (74.65), which suggests that the unmasked model generalises well with just two training domains.

The accuracy of the masked model trained on the laptop domain (77.82) is quite high as well, and significantly higher than the accuracy of the unmasked model trained on the same domain, indicating that for a single domain, the masking helps improve the generality of the model. However, the same is not true in the dual-domain, where the accuracy of the masked model drops to 75.00. This suggests that the unmasked model trained in a dual-domain has already reached its general-domain optimum (as proposed in Section 3.2.2), and therefore the masking only decreases its performance by hiding information.

Based on these results, we decide to use the unmasked model trained on the combined data for the sentiment analysis task in our ADA-X system.

Chapter 5

ADA-X System

In this chapter, we introduce the system architecture of our general ADA-X implementation, including an interactive front-end application based on the *Botplication* design principles proposed by Klopfenstein et al. [11]. We conclude by evaluating the performance and usability of our system.

5.1 Architecture

The full ADA-X system architecture diagram is shown in Figure 5.1. The diagram provides an example of a full pass of the system for a camera product, although ADA-X works for any product. We will refer to this diagram in the following sections.

5.1.1 Back-end

The back-end of the system consists of three distinct processes:

1. Ontology extraction,
2. QBAF extraction, and
3. a *conversational agent* that handles interaction with the front-end application.

Data flows on the diagram from left to right: the ontology for cameras is used to extract a QBAF for the Canon EOS 200D camera model, and the QBAF for the model is used in the conversational agent to communicate information about the model to the user via the front-end application.

While the conversational agent interacts with the user in real-time, the ontology and QBAF extraction processes run autonomously, and interact with the rest of the system only through their respective databases where they store the extracted data. This is the case for two reasons:

1. The ontology and QBAF extraction processes mine information from a large number of reviews, which takes a substantial amount of time. Therefore, implementing these processes in real-time would lead to unacceptable delays for the user, who expects fluid interaction with the system.
2. The extracted ontology and QBAF data is often re-used by multiple processes: the same ontology for cameras can be used to extract a QBAF for any camera model, and the same QBAF for a particular model can be used in conversations about the model with a number of different users. Therefore, extracting an ontology or QBAF from scratch each time would waste a lot of computational power.

Ontology extraction process

The ontology extraction process uses Amazon user reviews to extract ontologies for product categories with the method detailed in Chapter 3. As each ontology requires mining thousands of review texts (around 30,000 reviews was deemed sufficient in Section 3.2.1), extracting ontologies for each of the thousands of product categories on Amazon requires a large computational effort. However, once an ontology for a product category such as cameras has been extracted, there is

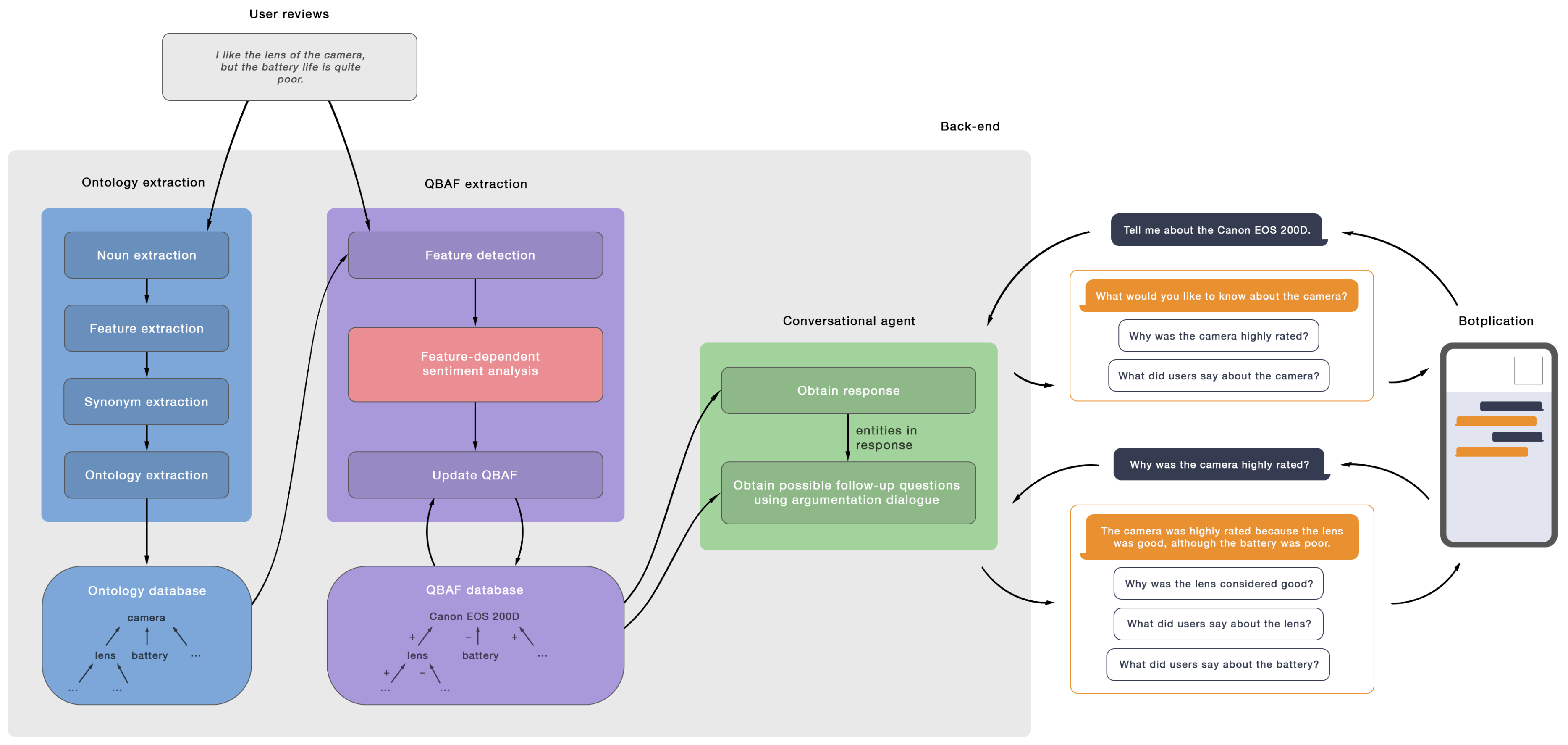

User reviews
I like the lens of the camera, but the battery life is quite poor.
Back-end
Ontology extraction
Noun extraction
Feature extraction
Synonym extraction
Ontology extraction
Ontology database
camera
lens
battery
QBAF extraction
Feature detection
Feature-dependent sentiment analysis
Update QBAF
QBAF database
Canon EOS 200D
lens
battery
Conversational agent
Obtain response
entities in response
Obtain possible follow-up questions using argumentation dialogue
Tell me about the Canon EOS 200D.
What would you like to know about the camera?
Why was the camera highly rated?
What did users say about the camera?
Why was the camera highly rated?
The camera was highly rated because the lens was good, although the battery was poor.
Why was the lens considered good?
What did users say about the lens?
What did users say about the battery?
Botplication

Figure 5.1: The full ADA-X system architecture, exemplified for a camera product

no need to update the ontology for a while assuming it is accurate. Although the composition or meaning of products can change over time, for most product categories, any changes usually happen slowly over the course of several years. Therefore, we propose that an ontology is initially extracted once for each product category, after which a background process can update the ontologies for categories with lots of new products when needed.

QBAF extraction process

The QBAF extraction process uses Amazon user reviews, the extracted ontologies, and the sentiment analysis method detailed in Chapter 4 to extract QBAFs for product models with the method detailed in Section 2.1. As for ontology extraction, the initial extraction of QBAFs for all Amazon products is a costly process. However, unlike for the ontology extraction, it is important that the QBAF for a product is updated for each new review, in order for the explanations to accurately reflect the entire set of reviews for the product. Therefore, the QBAF extraction requires a continuous background process. Note, however, that this background process is not as expensive as the initialisation of the QBAFs, as the computationally heavy tasks of feature detection and sentiment analysis will only have to be performed for the single new review instance.

Conversational agent

The conversational agent is responsible for the dialogue between the user and system. The user can initiate a conversation by requesting information about a particular product on the front-end application. The conversational agent then loads the QBAF for the product from the QBAF database, which it uses to direct the conversation. For each query from the user, the agent returns both a response for the query, as well as options for follow-up questions about the entities mentioned in its response. By only allowing the user to select from a pre-defined set of query options, the agent guides the conversation so that it stays in the familiar domain of the argumentation dialogue detailed in Section 2.1.5.

Multiple users can use the system at the same time, so the agent processes each request on its own thread in order to minimise the response time. The agent keeps track of the conversations by assigning each user a unique identifier.

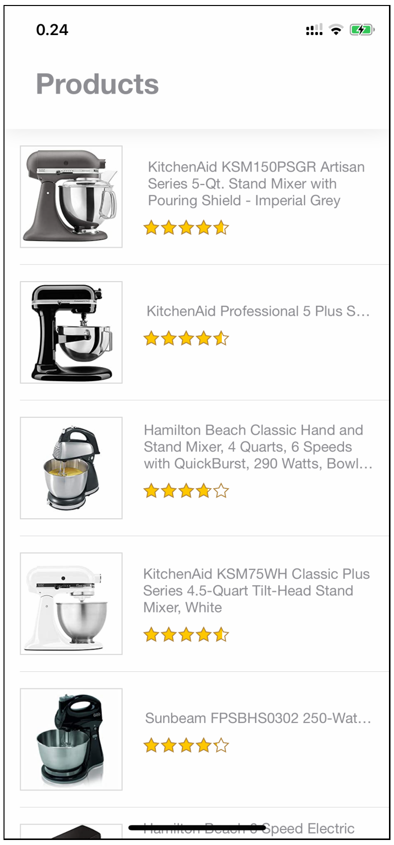
5.1.2 iOS Botplication

Based on the evaluation of conversational methods in Section 2.4.3, we chose to implement a Botplication front-end for ADA, which we call *ADABot*. To contrast with the conversational interface of ADABot, we also implement an app with a graph-based interface called *ADAGraph*. Both of the apps interact with the same back-end via a network connection. Figure 5.2 shows screenshots from the applications.

The first screenshot 5.2a shows a simple product selection screen, which the user uses to browse products in both of the applications. As our resources our limited, we cannot mine ontologies and QBAFs for the whole set of Amazon products, so the product selection screen displays only a small selection of products; a fully developed system would include a product search functionality.

The lower screenshots 5.2c and 5.2d are from the ADABot application. Once the user has selected a product they are interested in, ADABot initiates the conversation by asking the user what they would like to know about the product. The user can tap on any of ADABot's messages to reveal a set of possible questions determined by the argumentation dialogue. The subjects of these questions are the arguments mentioned in the message, which are highlighted in bold. For example, in 5.2c, the user can ask about either the mixer, the motor, or the bowl, for which two query options are presented.

An example of a short conversation between the user and ADABot is shown in 5.2d. The conversation starts from a general view of the reviewers' sentiment towards the product, and from there delves deeper into more specific aspects of the product by utilising its ontology. Through this conversation, the user not only gains a better understanding of why the product was highly rated, but possibly also discovers more about the importance of the various aspects of the product, which supports the *user revealment* property introduced in Section 2.4.3. To explore various aspects of the product, the user can at any point return to a previous point in the conversation by tapping on a previous message, which is one key advantage of the message-based Botplication design.



0.24

Products

KitchenAid KSM150PSGR Artisan Series 5-Qt. Stand Mixer with Pouring Shield - Imperial Grey

KitchenAid Professional 5 Plus S…

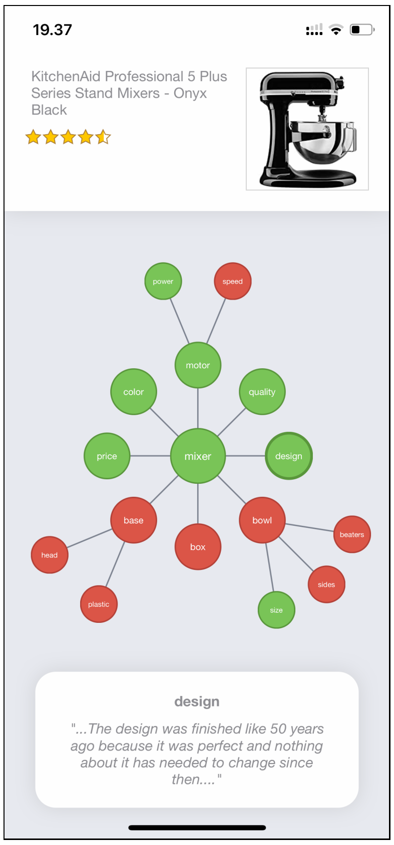
Hamilton Beach Classic Hand and Stand Mixer, 4 Quarts, 6 Speeds with QuickBurst, 290 Watts, Bowl…

KitchenAid KSM75WH Classic Plus Series 4.5-Quart Tilt-Head Stand Mixer, White

Sunbeam FPSBHS0302 250-Wat…

Hamilton Beach 6 Speed Electric

(a) Product selection screen



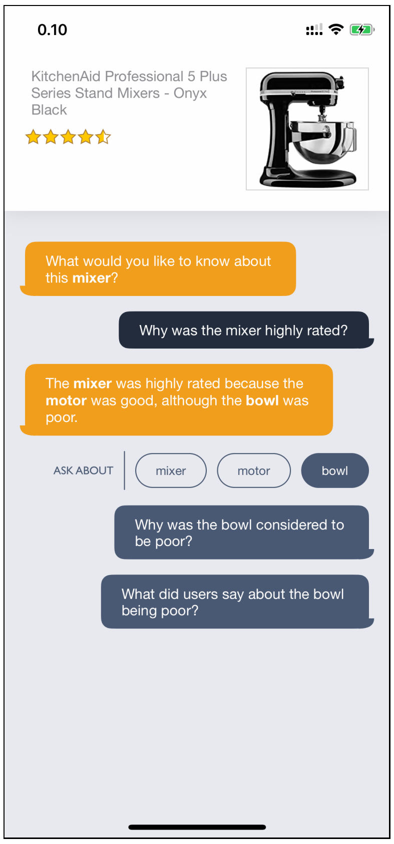
19.37

KitchenAid Professional 5 Plus Series Stand Mixers - Onyx Black

design

"...The design was finished like 50 years ago because it was perfect and nothing about it has needed to change since then...."

(b) ADAGraph



0.10

KitchenAid Professional 5 Plus Series Stand Mixers - Onyx Black

What would you like to know about this **mixer**?

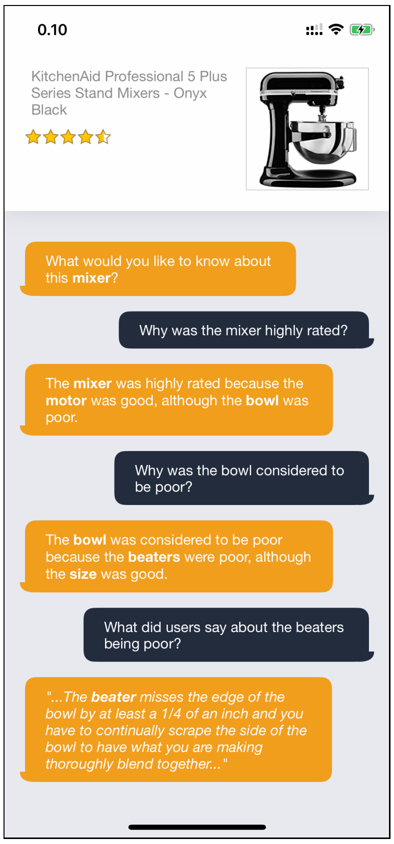
Why was the mixer highly rated?

The **mixer** was highly rated because the **motor** was good, although the **bowl** was poor.

Why was the bowl considered to be poor?

What did users say about the bowl being poor?

(c) ADABot user controls



0.10

KitchenAid Professional 5 Plus Series Stand Mixers - Onyx Black

What would you like to know about this **mixer**?

Why was the mixer highly rated?

The **mixer** was highly rated because the **motor** was good, although the **bowl** was poor.

Why was the bowl considered to be poor?

The **bowl** was considered to be poor because the **beaters** were poor, although the **size** was good.

What did users say about the beaters being poor?

"...The ***beater*** *misses the edge of the bowl by at least a 1/4 of an inch and you have to continually scrape the side of the bowl to have what you are making thoroughly blend together..."*

(d) Example of a conversation with ADABot

Figure 5.2: Screenshots from the ADA front-end applications

The user interface of ADAGraph is shown in screenshot 5.2b. Unlike ADABot, ADAGraph is non-conversational, instead allowing the user to freely investigate the product's various aspects. The positively viewed arguments are shown in green, while the negatively viewed arguments are shown in red. The user can tap on an argument to reveal a phrase from user reviews that exemplifies its sentiment, corresponding to the *"What did users say about __?"* query of ADABot.

5.2 Evaluation

We evaluate both the performance and the usability of our system. To investigate its performance, we measure and evaluate various quantitative and qualitative aspects including throughput, server response time, and scalability. To investigate the usability of our system, we carry out user testing with the two front-end applications.

5.2.1 Performance

Throughput

We evaluate the throughput of the ontology and QBAF extraction processes, as they make up the computationally heavy part of our system. We measure the average computation times for the various subtasks of the two systems, as well as the total average computation time. We use the standard dataset sizes of 30,000 reviews for the ontology extraction and 1,000 reviews for the QBAF extraction. All other computations were performed on an Intel Core i7-6700K 4.00GHz CPU except for the neural network passes, which were performed on a NVIDIA GeForce GTX 1080 GPU. For each of the CPU computations, we implemented multiprocessing across four CPU cores. The results are shown in Table 5.1 for the ontology extraction and Table 5.2 for the QBAF extraction.

We note that the ontology extraction takes a lot longer than the QBAF extraction, further supporting the decision to separate the two processes. In each of the two processes, the neural network passes take up a majority of the computing time, although the token processing tasks of noun extraction and feature detection take up a significant amount of time as well. The QBAF construction task takes a minimal amount of time compared to the rest of the QBAF extraction process, which supports our claim that updating the QBAF with new reviews is not a computationally heavy task.

Calculating the system throughputs using the total computation times and assuming constant review set sizes, we obtain 13.30 ontologies per hour and 423.23 QBAFs per hour.

Task	Time in seconds
Noun extraction	41.59
Feature extraction	171.88
Synonym extraction	3.45
Ontology extraction	53.67
Entire process	270.59

Table 5.1: Average computation times for the ontology extraction process with 30,000 reviews, extracting on average 18.4 features per ontology, with 0.448 synonyms per feature

Task	Time in seconds
Feature detection	0.676
Sentiment analysis	7.732
Review aggregation	0.097
QBAF construction	0.001
Entire process	8.506

Table 5.2: Average computation times for the QBAF extraction process with 1,000 reviews

Response time

We measure the response time of the server to the main requests from the front-end applications, and show the results in Table 5.3. As expected, loading the QBAF for a product from the database takes slightly longer (87.5 ms) than using it to respond to messages (11.5 ms). However, all response times are no more than 0.1 seconds, which is considered an acceptable response time in order for a real-time system to feel "instantaneous" for the user.

Application	Task	Time in ms
ADABot	Load product	87.5
	Reply to message	11.5
ADAGraph	Load graph	98.8

Table 5.3: Mean response times for the system's various interactive tasks

Scalability

With the throughputs calculated earlier in this Section, the ontology and QBAF extraction processes would take years to complete for the entire set of products on Amazon. However, these throughputs were calculated for only a single machine, where in reality the processes could easily be performed in parallel on hundreds of machines. This is due to the high intra- and inter-separability of the ontology and QBAF extraction processes: within the processes, reviews can largely be processed independently of each other, while processes for different products can be separated altogether. The separable nature of the processes suggests that the system is well-scalable through parallelisation.

5.2.2 User evaluation

We evaluate the explanation interfaces of ADABot and ADAGraph through a user trial. The goal of the trial is to evaluate the performance of two interfaces in explaining customer reviews to the user. Therefore, in addition to presenting users with product review explanations using the two interfaces, we include a third explanation method of showing the user the two most recommended customer reviews by Amazon for the product. The aim of this is to emulate the conventional user experience, where users read only a subset of the entire set of reviews. We will call this explanation interface TOP-2.

Our user trial is inspired by that of Gedikli et al. [7], who aimed to evaluate explanation types for recommender systems. The procedure is as follows:

Procedure 1 User trial for explanation methods

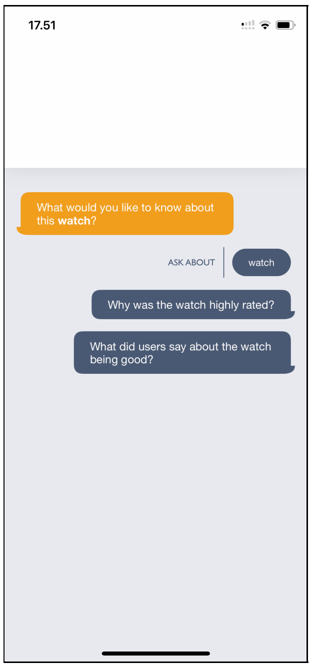
```
1: procedure USER TRIAL
2:     P = Set of products to explain
3:     E = Set of explanation interfaces
4:     for all (p, e) in randomly paired P×E do:
5:         Present product p with interface e to the user.
6:         Ask the user to rate p on a scale from 1 to 7.
7:     end for
8:     for all p in P do:
9:         Present Amazon page for product p to the user.
10:        Ask the user to rate p again on a scale from 1 to 7.
11:    end for
12:    Ask the user to rate the explanation interfaces.
13: end procedure
```

For our experiments, we use a product set P of six wrist watches due to their prevalence in every-day life. Each user is shown explanations for all six watches, which means that two products are shown with each of the three explanation interfaces. We use the above procedure to measure two aspects of the explanation interfaces:

1. **Effectiveness -** The effectiveness of an explanation interface can be defined as its ability to give the user an accurate image of the product. We measure effectiveness of the explanation interfaces by the difference in the ratings for a product given by a user using the interface (line 6 of the procedure), and after seeing more detailed information about the product on its Amazon page (line 10). The smaller the difference between the two ratings, the more effective we consider the explanation interface.

2. **Satisfaction -** We ask the users to rate their satisfaction with the three explanation interfaces at the end of the trial (line 12). This gives us an idea of the users' *perceived* effectiveness of the interface and the quality of its explanations.

When we present the explanation interfaces to the user (line 5), we reveal as little additional information about the product as possible. This includes hiding the product title, image, and rating from the ADABot and ADAGraph interfaces. The experimental interfaces are shown in Figure 5.3.

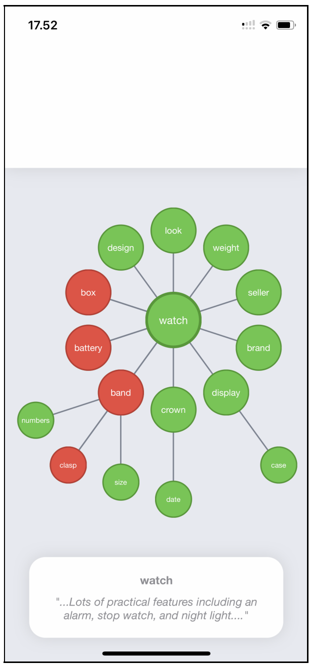


What would you like to know about this **watch**?

Why was the watch highly rated?

What did users say about the watch being good?

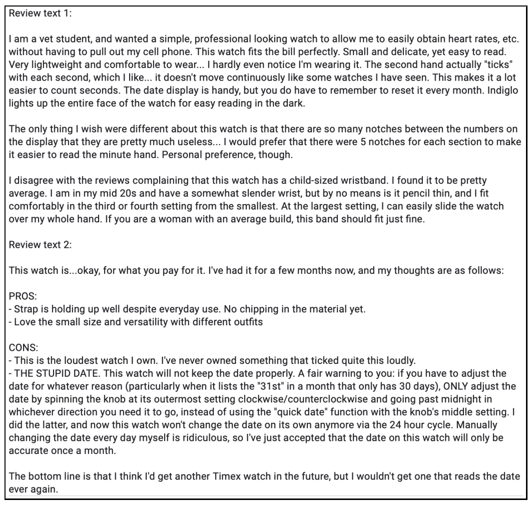
(a) ADABot



watch

"...Lots of practical features including an alarm, stop watch, and night light...."

(b) ADAGraph



Review text 1:

I am a vet student, and wanted a simple, professional looking watch to allow me to easily obtain heart rates, etc. without having to pull out my cell phone. This watch fits the bill perfectly. Small and delicate, yet easy to read. Very lightweight and comfortable to wear... I hardly even notice I'm wearing it. The second hand actually "ticks" with each second, which I like... it doesn't move continuously like some watches I have seen. This makes it a lot easier to count seconds. The date display is handy, but you do have to remember to reset it every month. Indiglo lights up the entire face of the watch for easy reading in the dark.

The only thing I wish were different about this watch is that there are so many notches between the numbers on the display that they are pretty much useless... I would prefer that there were 5 notches for each section to make it easier to read the minute hand. Personal preference, though.

I disagree with the reviews complaining that this watch has a child-sized wristband. I found it to be pretty average. I am in my mid 20s and have a somewhat slender wrist, but by no means is it pencil thin, and I fit comfortably in the third or fourth setting from the smallest. At the largest setting, I can easily slide the watch over my whole hand. If you are a woman with an average build, this band should fit just fine.

Review text 2:

This watch is...okay, for what you pay for it. I've had it for a few months now, and my thoughts are as follows:

PROS:
- Strap is holding up well despite everyday use. No chipping in the material yet.
- Love the small size and versatility with different outfits

CONS:
- This is the loudest watch I own. I've never owned something that ticked quite this loudly.
- THE STUPID DATE. This watch will not keep the date properly. A fair warning to you: if you have to adjust the date for whatever reason (particularly when it lists the "31st" in a month that only has 30 days), ONLY adjust the date by spinning the knob at its outermost setting clockwise/counterclockwise and going past midnight in whichever direction you need it to go, instead of using the "quick date" function with the knob's middle setting. I did the latter, and now this watch won't change the date on its own anymore via the 24 hour cycle. Manually changing the date every day myself is ridiculous, so I've just accepted that the date on this watch will only be accurate once a month.

The bottom line is that I think I'd get another Timex watch in the future, but I wouldn't get one that reads the date ever again.

(c) TOP-2

Figure 5.3: The experiment set-up for each of the interfaces, with all product information such as titles and ratings hidden.

We perform the trial with a group of three users. Due to COVID-19 restrictions, the users operate the applications through a video call via spoken commands, which might reflect on their user experience. After the procedure, we interview each participant about their opinions on the interfaces. The aggregated results for effectiveness and satisfaction are shown in Table 5.4.

Interface	Mean absolute rating difference	Mean satisfaction
ADABot	1.833	5.00
ADAGraph	1.500	4.67
TOP-2	1.667	3.00

Table 5.4: User interface trial results

We observe that ADAGraph obtains the lowest mean rating difference between the users' ratings using the interface and the users' ratings with complete information about the product. This suggests that ADAGraph is the most effective out of the three interfaces at providing the user with an accurate view of the product. In the interviews, two participants commented that ADAGraph's colourful node visualisation gave them a clear idea of the overall sentiment towards the product, which might have helped in this aspect. ADABot appears to be the least effective in this aspect; one participant commented that ADABot's dialogue does not reveal enough information about the product.

However, ADABot achieved the best performance in the satisfaction category. The participants seemed to be divided into two groups: one participant favoured ADAGraph due to the accessibility of its information, while two participants favoured ADABot's interface, as they thought its conversational interface was more readable. Based on the interviews, ADABot appeared to be most user-friendly out of the three interfaces. The plain review texts of TOP-2 performed worst in the mean satisfaction category; two participants commented that they felt like they got a biased view of the products from reading just two reviews. One participant also disliked having to go through the entire review texts to find the information that was relevant to them.

The results and interviews suggest that the explanations of ADABot and ADAGraph improve the user experience over simply reading the individual review texts. While ADAGraph provides more information which might lead to a more accurate image of the product, ADABot appears to be more user-friendly.

Chapter 6

Conclusion

In this project, we have extended the work of Cocarascu et al. [3] by proposing a general-domain conversational review explanation system, ADA-X, and implementing it as a server-client application with a user-facing messaging interface. The generality of ADA-X is provided by a novel domain-independent ontology extraction method using the state-of-the-art language model BERT, which we have shown to obtain ontologies with an F1-score multiple times higher than the state-of-the-art general knowledge networks of ConceptNet and WordNet. Furthermore, we have implemented a state-of-the-art feature-based sentiment analysis method, which we have shown to perform with near-optimal accuracy when applied in the general-domain despite being trained on domain-specific data.

ADA-X has been shown to be capable of providing dialogical feature-based explanations for any Amazon product, and its methodology could be extended to any review domain with sufficient amounts of data. Our user experiments have shown that the explanations provided by ADA-X improve the user experience of browsing product reviews, and that they can be communicated via different user interfaces for different purposes.

6.1 Future work

The focus of this project has been largely theoretical, as a large part of the project was spent experimenting with general-domain NLP. The theoretical aspects have left us little time to perfect ADA-X as a practical product. In this section, we propose some possible extensions to improve the practicality of ADA-X, as well as discuss its potential industry applications.

6.1.1 Extending the conversational aspect

The current argumentation dialogue of ADA-X is quite simple, relying on just two unique query templates in which arguments are embedded. One way to make the conversations with ADA-X more varied would be to increase the number of different queries in the dialogue. For example, one could easily extend the dialogue by adding the query *"What were the most polarising features of __?"*

Furthermore, the phrase examples given by ADA could be more effective if we could highlight the portion of the text where the sentiment is expressed. For example, in the relatively long review phrase

"...Bought this for my friend's daughter who was turning seven and it's great that this ***brand*** *makes beautiful and fun watches for young kids to learn to tell time..."*

it would be helpful if we could highlight *"this brand makes beautiful and fun watches"*, as this portion is directly related to the sentiment towards the *brand* aspect. Due to the positional token embeddings of the Transformer encoder, this could in theory be possible by observing the attention values between each of the tokens and the *brand* token. However, the explainability of the attention mechanism is still an open debate: while Jain et al. claimed in early 2019 that *Attention is not explanation* [9], later that year Wiegreffe et al. responded with a paper titled *Attention is not not explanation* [26].

6.1.2 Recommender system

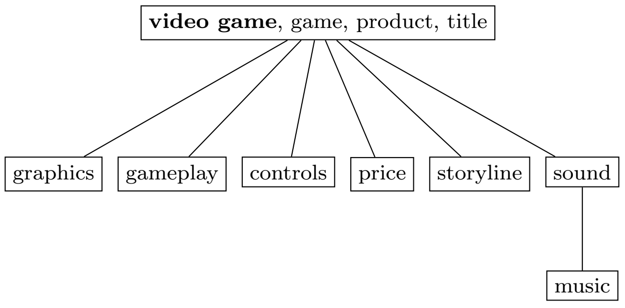
Perhaps the most practical application of ADA-X would be within a recommender system, where it could provide the user with feature-based explanations for why a product was recommended to them. It has been shown that such explanations can help improve the effectiveness of a recommender system [7].

But rather than just basing its explanations on the features of products, an entire recommendation system could be built on the feature-based representations that ADA-X is capable of extracting. To this day, Amazon's recommendation system is still largely based on *Item-to-Item Collaborative Filtering* [14], a method invented at Amazon in 1998. This method is based on correlations between products in terms of customers' purchase histories, and although focusing on such a singular view of products used to make sense in terms of computational costs, high-performance computing has come far since twenty years ago.

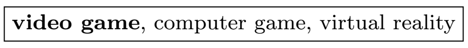
The focus on products as singular entities ignores any feature-level preferences of users: for example, one group of people might be more interested in the *design* of items, while a more practically-minded group might look for *durability*. Furthermore, unlike products, many features are relevant across several product categories; a feature-based recommendation system could take advantage of such inter-categorical connections. For these reasons, we propose that a *feature-to-feature* approach to collaborative filtering using the feature-based product representations that ADA-X is capable of extracting could improve the accuracy of Amazon's recommender system.

Appendix A

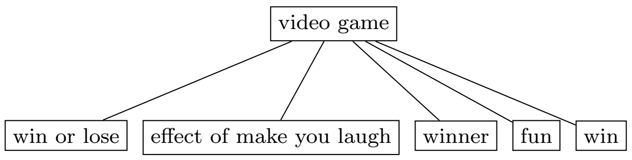
Extracted ontologies



(a) Extracted ontology

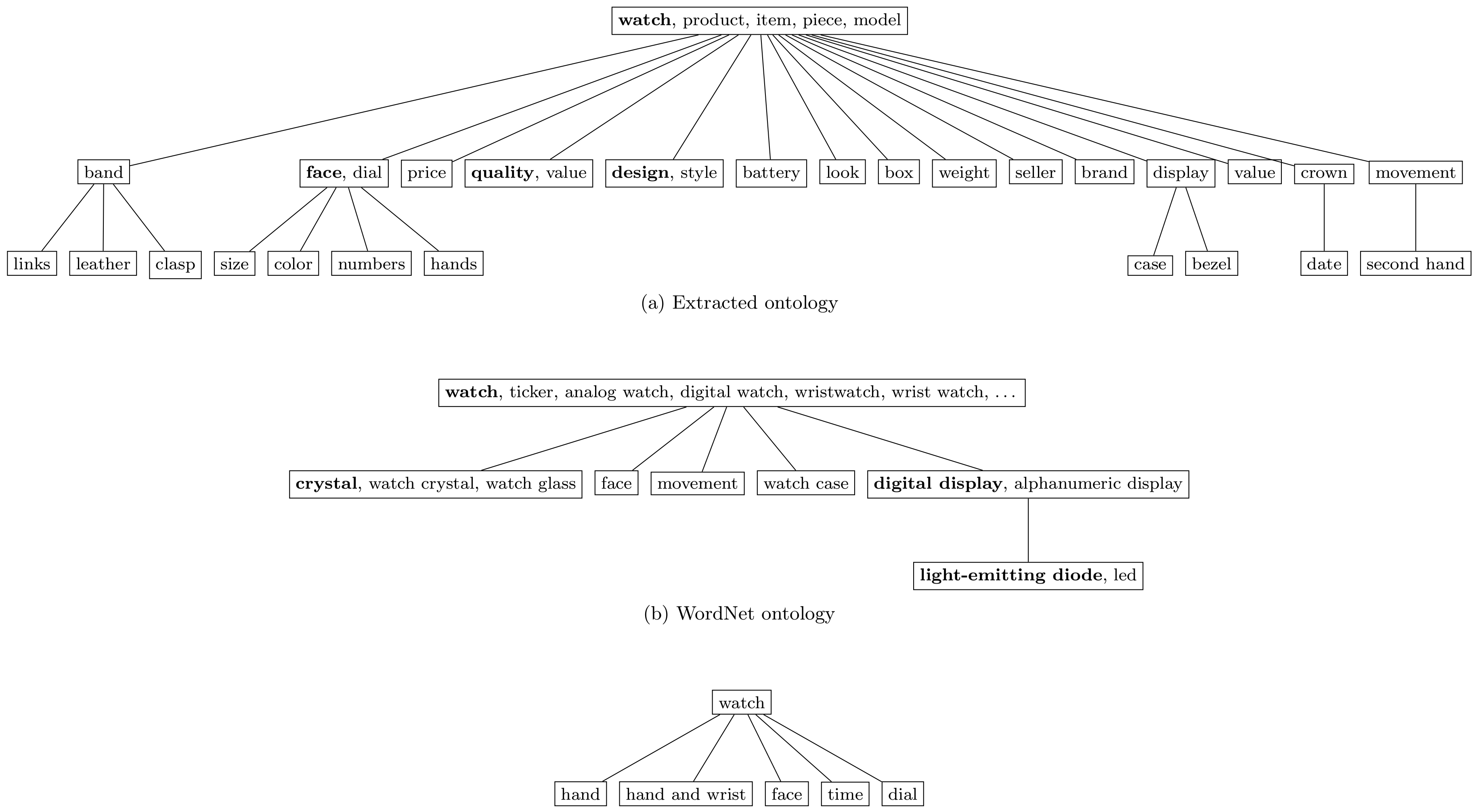


(b) WordNet ontology



(c) COMeT ontology

Figure A.1: Full ontologies for *video game*

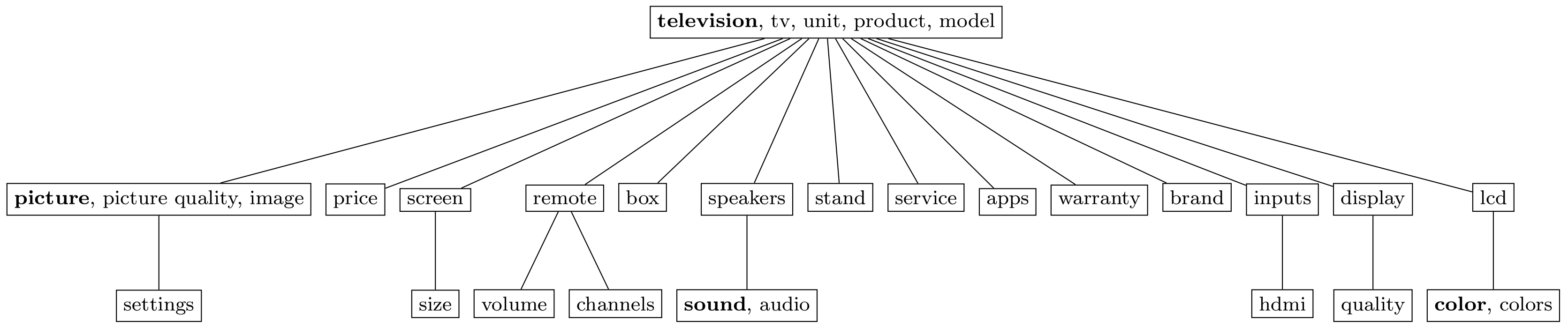


(a) Extracted ontology

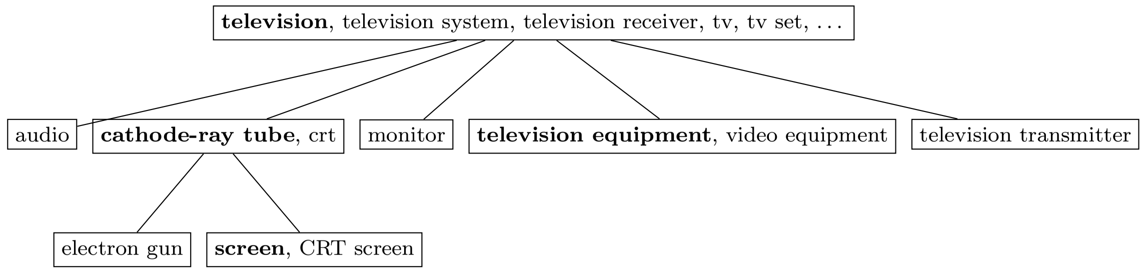
(b) WordNet ontology

(c) COMeT ontology

Figure A.2: Full ontologies for *watch*

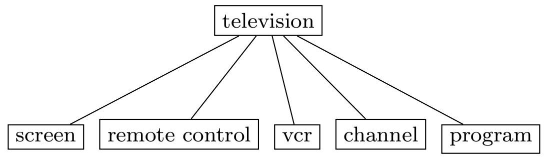


(a) Extracted ontology



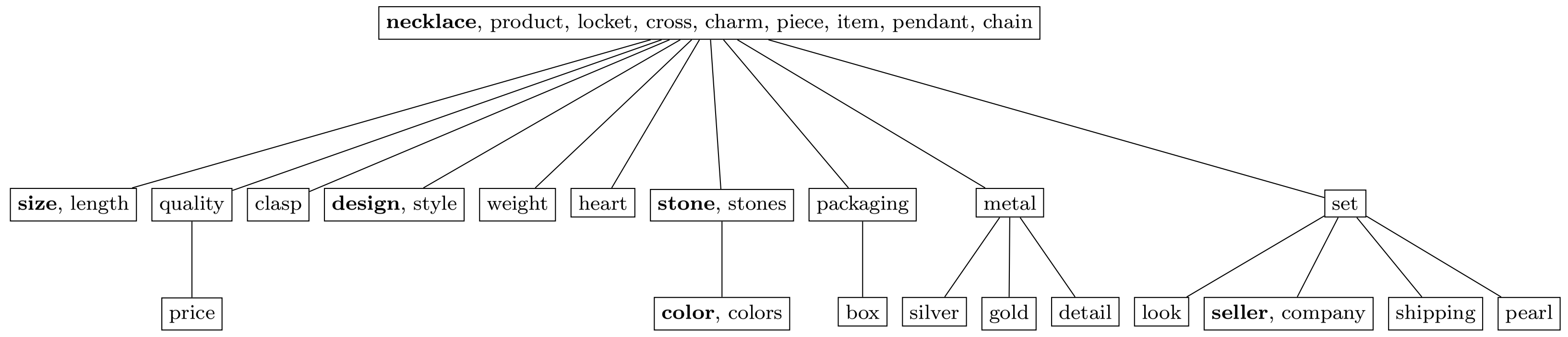
television, television system, television receiver, tv, tv set, . . .
audio
cathode-ray tube, crt
monitor
television equipment, video equipment
television transmitter
electron gun
screen, CRT screen

(b) WordNet ontology

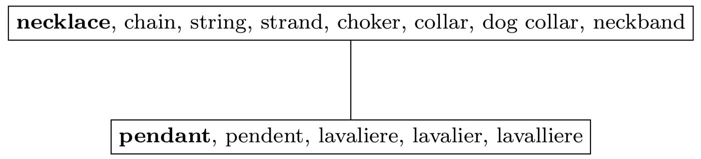


(c) COMeT ontology

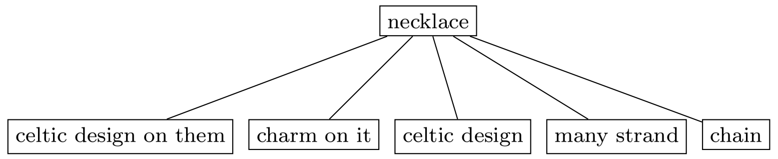
Figure A.3: Full ontologies for *television*



(a) Extracted ontology

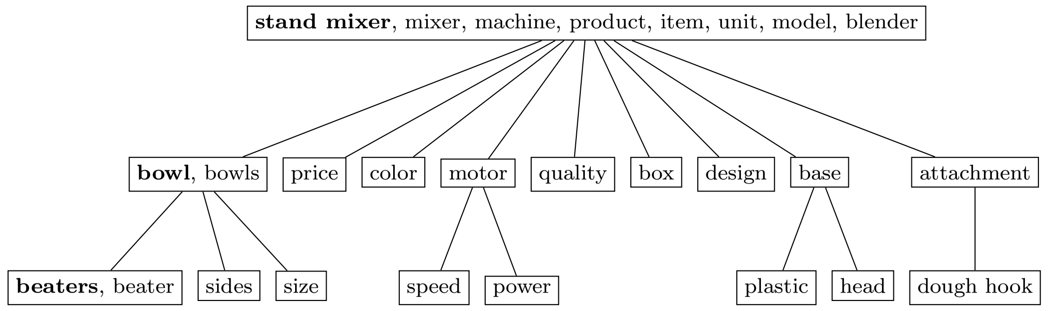

necklace, chain, string, strand, choker, collar, dog collar, neckband
pendant, pendent, lavaliere, lavalier, lavalliere

(b) WordNet ontology

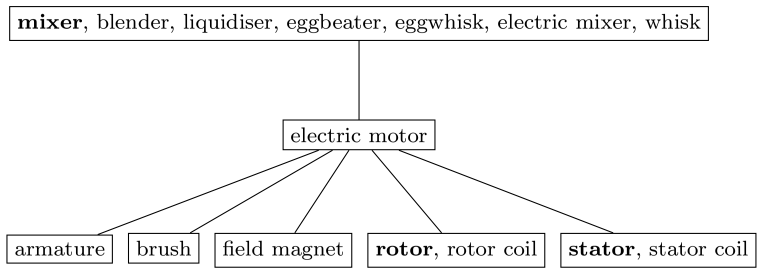


(c) COMeT ontology

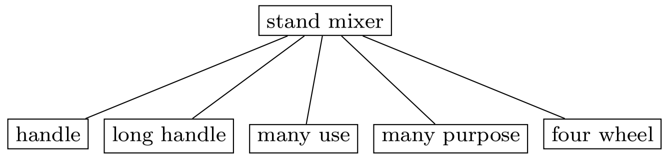
Figure A.4: Full ontologies for *necklace*



(a) Extracted ontology



(b) WordNet ontology*



(c) COMeT ontology

Figure A.5: Full ontologies for *stand mixer* (*the term *stand mixer* returned no results in WordNet, so *mixer* was used instead)

Bibliography

[1] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*, 2019.

[2] O. Cocarascu, A. Rago, and F. Toni. From formal argumentation to conversational systems. ACM, Jun 1, 2019.

[3] Oana Cocarascu, Antonio Rago, and Francesca Toni. Extracting dialogical explanations for review aggregations with argumentative dialogical agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 1261–1269, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Ying Ding, Jianfei Yu, and Jing Jiang. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3436–3442. AAAI Press, 2017.

[6] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-dependent sentiment classification with bert. *IEEE Access*, 7:154290–154299, 2019.

[7] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should i explain? a comparison of different explanation types for recommender systems, 2014. ID: 272548.

[8] R. Chulaka Gunasekara, David Nahamoo, Lazaros C. Polymenakos, David Echeverr ' ia Ciaurri, Jatin Ganhotra, and Kshitij P. Fadnis. Quantized dialog - a general approach for conversational systems. *Computer Speech & Language*, 54:17–30, 2019.

[9] Sarthak Jain and Byron C. Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[10] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160,, Portland, Oregon, USA, jun 2011. Association for Computational Linguistics.

[11] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems, DIS '17, Edinburgh, United Kingdom, June 10-14, 2017*, pages 555–565, 2017. DBLP:conf/ACMdis/2017.

[12] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[13] Artuur Leeuwenberg, Mihaela Vela, Jon Dehdari, and Josef van Genabith. A minimally supervised approach for synonym extraction with word embeddings. *The Prague Bulletin of Mathematical Linguistics*, 105(1):111–142, 2016.

[14] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[15] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA, 2013. Association for Computing Machinery.

[16] Mary L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[19] George A. Miller. Wordnet: A lexical database for english. *Commun.ACM*, 38(11):39–41, nov 1995.

[20] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

[21] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orph 'ee De Clercq, V. 'eronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud Mar ' ia Jim 'enez Zafra, and G ul csen Eryi ugit. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30,, San Diego, California, jun 2016. Association for Computational Linguistics.

[22] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Comput.Linguist.*, 37(1):9–27, mar 2011.

[23] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17, page 117–126, New York, NY, USA, 2017. Association for Computing Machinery.

[24] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press, 2017.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[26] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

[27] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdiscip.Rev.Data Min.Knowl.Discov.*, 8(4), 2018.