Contents

1 Introduction 2

2 Animacy and annotation 4
   2.1 Grammatical effects of animacy 6
   2.2 Probabilistic effects 9
   2.3 Animacy in natural language processing 12
   2.4 Annotation schemes 13
      2.4.1 Referentiebestand Nederlands (RBN) format 13
      2.4.2 Stanford-Edinburgh paraphrase project format 15

3 Related work 17
   3.1 Animacy classification based on morphosyntactic corpus frequencies 18
   3.2 Animacy identification using lexical-semantic databases 22
   3.3 Semi-automatic labeling using a dictionary 24
   3.4 Animacy knowledge discovery from web-scale N-grams 24
   3.5 Animacy classification by sparse logistic regression 25

4 Data and methodology 28
   4.1 Classification task 28
   4.2 Data 30
   4.3 Context data 32
   4.4 Memory-based learning 33
      4.4.1 Training and testing 35
      4.4.2 Similarity measures 37
      4.4.3 Feature relevancy 39
      4.4.4 Class voting 40
4.4.5 Wrapped Progressive Sampling .......... 41
4.5 Evaluation method .......................... 42
4.5.1 Baseline .................................. 43

5 Features .................................. 45
5.1 Object/subject frequency of verb relation .... 46
5.2 Distributional lexical features .................. 47
  5.2.1 Feature value ............................. 48
5.3 Verb subject relations ........................ 52
5.4 Verb object relations .......................... 54
5.5 Adjective relations ............................ 55

6 Evaluation ................................ 57
6.1 Features .................................... 57
6.2 Best settings ................................. 58
6.3 Noun frequency ............................... 60
6.4 Two-way classification ........................ 62
6.5 Data balancing ................................. 63

7 Applications and future work ..................... 65
7.1 Corpus annotation ............................. 65
7.2 Languages and resource dependence .......... 66
7.3 NLP tasks .................................. 67
7.4 Classifier improvement ........................ 68

8 Conclusion ................................ 69
List of Tables

4.1 Selected examples of animacy categorizations in the Cornetto lexical-semantic database ......................................................... 31
4.2 Example of extracted subject dependency relations .................. 33
4.3 Example feature vector for KNN animacy classification ............ 35
4.4 Example feature vector of a novel word for KNN animacy classi-
     fication ........................................................................ 36
4.5 Example of a confusion matrix ............................................. 43

5.1 Object/subject ratio feature performance ............................ 47
5.2 Comparison of feature value metrics .................................... 49
5.3 An example contingency table for association measure computa-
     tion ........................................................................... 51
5.4 Classifier performance for different numbers of subject features . 53
5.5 Most informative verbs for subject dependencies .................. 53
5.6 Most informative verbs for subject dependencies, larger feature set 54
5.7 Classifier performance for different numbers of object features . 54
5.8 Most informative verbs for object dependencies ..................... 55
5.9 Classifier performance for different numbers of adjective features 56
5.10 Most informative verbs for adjective dependencies ............... 56

6.1 The best classifier results of each dependency feature type ...... 58
6.2 Classifier results for different object/subject/adjective feature proportions ............................................................. 58
6.3 Performance of a classifier with optimal settings .................. 58
6.4 Confusion matrix of the classifier with optimal settings ........ 60
6.5 Classifier performance of Øvrelid (2009) for different frequency
     bins ............................................................................. 61
6.6 Classifier performance on datasets with different frequency cutoffs 62
6.7 Performance of two kinds of two-way animacy classifiers . . . . . . 62
   (a) Animate-Inanimate ........................................... 62
   (b) Human-Nonhuman .............................................. 62
6.8 Confusion matrices of two kinds of two-way animacy classifiers . 63
   (a) Animate-Inanimate ........................................... 63
   (b) Human-Nonhuman .............................................. 63
6.9 Classifier performance on a class-balanced data set ............... 63
6.10 Confusion matrix of the classifier with class-balanced data .... 64
Abstract

We present an automatic animacy classifier for Dutch that can determine the animacy status of nouns — how alive the noun’s referent is (human, inanimate, etc.). Animacy is a semantic property that has been shown to play a role in human sentence processing, felicity and grammaticality ("the spoon *who is on the table fell."). We expect knowledge about animacy to be helpful for parsing, translation and other NLP tasks, although animacy is not marked explicitly in Dutch.

Only a few animacy classifiers and animacy-annotated corpora exist internationally. For Dutch, animacy information is only available in the Cornetto lexical-semantic database. We augment this lexical information with context information from the Dutch Lassy Large treebank, to create training data for an animacy classifier that uses context features.

An existing Swedish animacy classifier (Överlid, 2009) uses the k-nearest neighbour algorithm with morphosyntactic distributional features, e.g. how frequently the noun occurs as a sentence subject in a corpus, to decide on the (predominant) animacy class. For Dutch we use the same algorithm, but with distributional lexical features, e.g. how frequently the noun occurs as a subject of the verb ‘to think’ in a corpus. The size of the Lassy Large corpus makes this possible, and the higher level of detail these word association features provide, increases the classifier accuracy and provides us with accurate Dutch-language animacy classification. These results allow (semi-)automatic corpus animacy annotation for creating animacy training resources, which can help other Dutch NLP tools to incorporate the animacy property of nouns.
Acknowledgements

I would like to thank everyone who helped me in writing this thesis with their ideas, discussions, suggestions and examples — in particular, my supervisor Dr. Gosse Bouma, who initially gave me the idea of trying animacy classification for Dutch, helped in obtaining the necessary data, and checked many drafts throughout. I would also like to thank Prof. Gertjan van Noord for reviewing the thesis, and Dr. Lilja Øverlid for interesting discussions on the topic. I am grateful to the Alfa-informatica department for supporting this research and providing access to data and resources.

I would also like to thank Ana Bosnic, Eliza Margaretha, Ihor Tytyk, Laura Handjojo, Sibel Ciddi and Victor Santos for their input, linguistics help, and their support. For checking language-specific examples, I am thankful to Willeke Bloem and Ye Tao. Thanks also go out to Anne van de Wetering, Lili Szabó and Michaela Regneri for various corrections and suggestions. Lastly, I would like to thank all the other LCT and Coli people I met during my master studies for always being supportive and making my master studies a much more interesting time!
Chapter 1

Introduction

In recent years, the animacy property of nouns has been shown to be a relevant one for natural language processing. It plays a role in various linguistic phenomena across languages, and can be used in determining sentence acceptability and grammaticality. However, animacy is rarely included in annotation efforts of text corpora and, perhaps for that reason, Natural Language Processing (NLP) tools rarely incorporate animacy in their algorithms. Automatically determining the animacy of nouns would allow NLP tools such as parsers to use this property, and allow animacy effects to be studied computationally with large amounts of data. Øvrelid (2009) has created an animacy classifier for the Swedish language, showing accuracy improvements in parsing, but it is specific to Swedish. We are not aware of any such classifier for Dutch.

In this thesis, we attempt to solve the problem of animacy classification for Dutch. Automatic animacy classification is the task of deciding which of several animacy-related semantic categories a noun belongs to. We will explore the phenomenon of animacy in language, which is more complicated than simply animate or inanimate. There is a wide variety of possible animacy classes, and their borders may be different for different grammatical phenomena. One can also debate whether animacy is a set of classes, a hierarchy, or even a gradient scale. Practical matters also play a large role in this problem, such as the availability of classifier training resources. For supervised learning, some sort of ‘gold standard’ animacy information is required. Several animacy classifiers have been developed for other languages than Dutch, though they all differ in their method, largely for practical reasons as well. We will examine them and discuss whether aspects of their methods are suitable for the Dutch language situation.

We will develop an animacy classifier for Dutch, using proven existing methods where possible, and implementing new ideas where it is practical. The goal is to have a system that can use the linguistic resources that are available for Dutch to provide the best possible classification, given the limitations of the resources. In the future, this classifier can aid in the annotation of Dutch corpora with animacy information, and help NLP tools for the Dutch language to use the animacy property of nouns.
Chapter 2 starts with a discussion of animacy, its possible categorizations, and the state of animacy in natural language processing and corpora annotation. Then, we will discuss existing approaches to animacy classification in chapter 3. We proceed to describe our methodology and the linguistic resources that we use in chapter 4. The linguistic features that we base our classification on are discussed in chapter 5, followed by a general evaluation of our system in chapter 6. We will end with a discussion of applications of our method, as well as possible future work in chapter 7 and conclude in chapter 8.
Chapter 2

Animacy and annotation

Animacy is a semantic property of nouns, that describes whether the referent of the noun is alive or sentient, and to what degree. It may also distinguish between various kinds of sentience. The most basic distinction is between animate and inanimate nouns. The animate category of nouns can include personal pronouns, person names, or words such as sister, participant, carpenter, dude, northerner and possibly cat, angel and dragon. Inanimate nouns can include fountain, second, observation, and possibly community, oak, and robot. Various categorizations and category boundaries are used in linguistic theory and found in languages. In this chapter, I will discuss animacy, its possible categorizations and its grammatical effects, and then I will show how the animacy property can be described or annotated in linguistic resources.

Semantically, animacy can be seen as a hierarchy, ranging from a reference to a human (most animate) to a noun that refers to something inanimate. Various categories and subcategories can be found in between, though there is always debate about what the category divisions should be. They differ by language, and may change over time. A basic example of such a hierarchy, which first appeared in Silverstein (1976), is HUMAN > ANIMAL > INANIMATE. In cases where animacy plays a role in a linguistic phenomenon, the phenomenon may apply only to elements above a certain cut-off point in this hierarchy, for example, only to nouns referencing animals or higher animate beings (de Swart et al., 2008). This kind of two-way distinction seems to be the most common form in which animacy affects grammar. Effects that cannot be explained by a two-way distinction are generally probabilistic or processing effects. I will discuss some examples of grammatical effects from different linguistic studies in the next section, and will discuss processing effects afterwards.

Some sources also include personal pronouns or personal names in the hierarchy, including them as strongly animate nouns. One such hierarchy is discussed in DeLancey (1981) to explain ergative case markings in some languages. The categories are: 1st & 2nd person > 3rd person > human > animate > natural forces > inanimate. In this type of case marking, the cut-off point between the use of two kinds of marking may lie between 1st & 2nd person and 3rd person pronouns. However, it has also been argued that the 'person'
property should not be conflated with animacy (Comrie, 1989).

de Swart et al. (2008) state that the animacy hierarchy should be seen as a gradient, based on the unclear boundaries of categories. In some cases, as will be shown in the next section, there are ‘grey areas’ when animacy categories are observed in language, where nouns that are borderline animate or inanimate may behave in both ways. However, such a view is problematic for the study of animacy effects on grammar, where rules that operate on categories are often used. It does lend itself to probabilistic accounts of animacy.

For some languages the animacy categorization has been found to be partly grammatical, as well as being semantic. This ‘grammatical animacy’ therefore also includes some nouns denoting objects and abstractions (Aissen, 1997). This seems to have originated through analogy and convention, and can be compared to grammatical gender which often doesn’t correspond to biological genders of noun referents. I will discuss some examples of this in the next section.

More elaborate animacy hierarchies, grounded in semantics, have also been used to describe the property. These can often be found in language documentation efforts rather than linguistic theories or grammars, such as the one used for the Cornetto lexical-semantic database (Martin et al., 2005), which subdivides the inanimate category into various subcategories. Such hierarchies aim to describe the semantics of animacy, rather than account for some grammatical effect it may cause, so they tend to be more elaborate. These semantic annotation schemes will be discussed in section 2.4.

Metaphors and expressions complicate matters: in figurative language it can be unclear what the referent is (Zaenen et al., 2004). In fictional narratives, a normally inanimate entity may be sentient, behaving more like an animate actor. de Hoop (2012) has studied this, using a Dutch book where the first person narrator is a painting, and comparing it to a book by the same author with an animate narrator. Preliminary results indicate that these sentient inanimate objects behave the way animate entities would behave in language. This shows that animacy is also context-dependent - in some cases, like metaphors or fictional narratives, sentience of entities can deviate from reality and this seems to affect the way they are processed as well, showing that animacy is based on semantics even though it can affect a language’s grammar.

Dahl and Frånvarud (1996) provide a non-exhaustive overview of ways in which animacy can affect grammar. They list the following:

- Subject and object marking (such as accusative case marking)
- NP-internal case markings (such as the possessive)
- Restriction of transitive subjects (requiring them to be more animate)
- Hierarchical restrictions (the subject needs to be more animate than the object)

In the next section, I will discuss some of these effects, and other cases where animacy affects grammar (section 2.1). After that, I will discuss processing effects of animacy — probabilistic tendencies that cannot be captured in rules,
but can show up in a statistical analysis (section 2.2). Then I will discuss what
the field of natural language processing has done with animacy (section 2.3), and
lastly, I will discuss some annotation schemes that have been used to capture
the semantic property of animacy in corpora (section 2.4).

2.1 Grammatical effects of animacy

There are many ways in which the animacy property can affect grammar in
a language. Along with the list of Dahl and Fraurud (1996), pronouns are
another phenomenon in which animacy can be involved. In English, the choice
of pronouns may be governed by animacy. The relative clause pronoun which
may only be used to refer to inanimate subjects, while who should be used for
animate subjects:

(1) a. The spoon which is on the table is mine.
   b. *The man which is sitting on the table is my friend.

(2) a. *The spoon who is on the table is mine.
   b. The man who is sitting on the table is my friend.

In different languages, different animacy classifications have been observed
and used to explain grammatical phenomena. For the English relative clause
pronouns above, a basic two-way distinction is sufficient to explain the observa-
tions, which roughly seems to match a standard animate-inanimate distinc-
tion, although around the cut-off point in the hierarchy (the animate-inanimate
border), both options seem to be used (examples from the British National
Corpus, Consortium et al. (2007)):

(3) One person holds the lead and stands behind the dog who is sitting.
(4) (...) the dog which was allowed to bark in the night, (...)

This shows that the notion of a ‘cut-off point’ may be problematic, and that
it should not be seen as a hard cutoff. There may be overlap between the
categories, and it is an example of a ‘grey area’ as discussed earlier. However,
for the following discussion of classifications in different languages, we will keep
using this terminology, since it is frequently used in the literature.

Some instances of grammatical effects of animacy require more than two
categories to be fully accounted for. A commonly used distinction, derived from
the original animacy hierarchy described in the previous section, distinguishes
human, (other) animate and inanimate referents of nouns. An account
of object agreement in four African languages uses these three categories as
syntactic agreement conditions (Woolford, 1999). In these languages, object
agreement is more likely to occur for objects that are either animate or human,
as opposed to inanimate. However, none of these languages use more than
two categories at once, so in this case there is still a single cut-off point or
category border, its exact location in the hierarchy just differs between related
languages. The authors can therefore model animacy as a binary feature, such
as +animate or +human.
Animacy seems to be partly grammatical. As well as nouns one would expect semantically, this ‘grammatical animacy’ may also include some nouns denoting objects and abstractions (Aissen, 1997). This becomes more clear if we look at other languages than English. Algonquian is an example of a family of languages in which the animacy categorization clearly doesn’t match what an ontology would consider to be animate or inanimate. A survey by Quinn (2001) of the Algonquian language Penobscot discusses these distinctions. Some examples of nouns that are unexpectedly considered animate are nouns for fluid containers such as kettle, cup and spoon, and written symbols like glyph, dice and playing card. The authors theorize that such semantic groups have come to be considered animate or inanimate by analogy to other words, though there are many exceptions that cannot be explained by such a theory. Another example is that the language considers some fruits as animate, and others as inanimate. Animate fruits are apple, blackberry and plum, while inanimate fruits are lemon, banana and cranberry. The authors theorize that the animates are a semantic group of softer or bigger fruits, while the inanimates are tougher or smaller. A similar distinction is observed with baked goods and grain products. It seems that, while this language does not follow the same animacy categorizations as an English ontology would, there is a logic to them. These categories are used in the grammar in this way, even though they do not match common biological definitions of animacy.

Quinn (2001) also note the existence of dual animates in this language, words that occur as both an animate and inanimate with different senses for each, and that of variable animates, which can be used in both ways with no difference in meaning. This is another example of a ‘grey area’ between categories as was discussed earlier.

In Dutch or English, there seem to be very few phenomena where animacy is explicitly marked or used in the grammar. But in some languages, the animacy category of nouns is clearly marked. A common case, also mentioned in the list of Dahl and Fraurud (1996), is in the marking of the grammatical case. In Russian, the animacy class of nouns is reflected in their accusative case marking. This marking distinguishes two animacy classes, animate and inanimate. Animate accusative nouns are marked in the same way as the genitive, and inanimate accusative nouns are marked in the same way as the nominative. This example from Fraser and Corbett (1995) demonstrates the difference:

5) pervogo (acc=gen) studenta (acc=gen)
   first student
   ‘the first student’

6) pervyj (acc=nom) zakon (acc=nom)
   first law
   ‘the first law’

The agreement of the adjective shows the same animacy pattern. Example 5 shows the marking on an animate noun and its adjective, example 6 shows an inanimate noun with the same adjective. Animacy affects case marking in different ways in a variety of other languages as well (Malchukov, 2008).
One of the few grammatical animacy effects present in Dutch is the selection of relative pronouns, somewhat similar to the English examples 1 and 2, though seemingly limited to the case of *wh*-cleft constructions, a type of construction that occurs in a sentence in which a particular constituent is put into focus by putting it in a dependent clause at the start of the sentence. In other contexts, just noun gender is sufficient to explain the selection of Dutch relative pronouns, but in the case of *wh*-clefts, animacy also needs to be involved:

(7) a. **Wat** ik leuk vind, is die **tafel**\(^{\text{gen=comm,-animate}}\) is that table

b. **Wat** ik leuk vind, is **dat** **huis**\(^{\text{gen=neut,-animate}}\) is that house

c. **Wie** ik leuk vind, is **dat** **kind**\(^{\text{gen=neut,+animate}}\) who is that child

d. **Wie** ik leuk vind, is die **vrouw**\(^{\text{gen=comm,+animate}}\) is that woman

These constructions occur in English as well, and can be phrased in a similar way. This example shows that in this construction, the relative pronoun does not vary only with gender, as it would if we would translate examples 1 and 2 to Dutch directly, using *d*-pronouns (*die, dat*). The animacy property is required to explain the variation in this example, just as in the English equivalents. For a more extensive analysis of this phenomenon that also includes gender effects, see van Kampen (2007). This example (7) is a constructed example based on their work. I would say that the third sentence needs a question mark, and that *wat* could also be used there. However, a corpus search\(^1\) shows almost no cases of *wat* being used to refer to animate entities, only some borderline exceptions such as:

(8) *Wat* men nu gedood of gevangen had, vormde maar een vijfde

What they now killed or captured had, constituted only a fifth

van de primaire mobilisatie van het Rode Leger

of the primary mobilisation of the Red Army

In this case, *wat* refers to a mobilisation of troops, which is a borderline case for animacy. Other words that were referenced by *wat* in *wh*-clefts in this data are *onkruid* (weeds), and *de IT'er* (the IT worker, as a concept). I found no convincing cases of animate referents, confirming the findings of van Kampen (2007).

Another example of animacy in Dutch, that also seems to require a different analysis than the basic animate-inanimate distinction, is provided in de Swart et al. (2008). In written Dutch, some quantifiers such as *meeste* ‘most’ and *beide* ‘both’ are marked with a suffix -*n* when they have a human referent (example 9) but are unmarked in reference to other entities (example 10):

---

\(^1\)The corpus was a dump of Wikipedia from 04-08-2011, automatically parsed with the Alpino parser.
The Dutch language lacks clear cases of animacy marking, such as the Russian example (5, 6). However, not all animacy effects are explicit in the grammar. Sometimes they are merely preferences, or processing effects. These effects have been studied in psycholinguistic literature. I will discuss some examples in the next section.

2.2 Probabilistic effects

All the animacy effects we have examined so far involved animacy categories, where one category had some effect and the other category had some other effect in a language. This is the way in which many linguistic theories view language, however, there are language effects that cannot be captured by such rules and categories. Some effects are better modeled with probabilities or exemplar-based models of language. Broadly speaking, rather than using binary grammatical rules that either do or do not apply, given a context, the ‘rules’ of probabilistic grammars can have a certain probability of being applied, given a context. This allows the modeling of tendencies as well as strict rules.

For animacy, one such probabilistic effect was found in the syntax of constructions with give in New Zealand and American English (Bresnan and Hay, 2008). A statistical model was used to predict the grammar of semantically similar but syntactically different phrases involving give for US English. They also include a model trained on NZ English. The phenomenon that they studied is called the dative alternation. It has been extensively studied in psycholinguistics, and it is often cited as an example of a syntactic difference without a meaning difference. For that reason, linguists have been studying why people choose one or the other to express the same thing. A transitive verb such as give can be phrased as a double object construction:

(11) He gave his friend the ticket.

Or as a prepositional dative:

(12) He gave the ticket to his friend.

The choice between these syntactic structures is considered to be the dependent variable in the model of Bresnan and Hay (2008), and different features such as definiteness and animacy of the referent are predictor variables (or features). This logistic regression model is then used to predict how likely each syntactic construction is to be used, given the predictor variables. For animacy, they use the categorization human - animal - other animate - inanimate. Using
this model setup, they find that give with an inanimate recipient is phrased in a double object construction significantly more often in NZ English than in US English, where the alternative, the prepositional dative, is more often used. In an earlier study, it was also found that inanimacy of the recipient in US English alone has a strong correlation with use of the prepositional dative (Bresnan et al., 2007), and including other features, they are able to predict this syntactic choice in US English correctly with 94% accuracy. This shows that the predictor variables used by the model indeed predict the data, and it also shows that animacy influences the use of the prepositional dative, even though there is no hard rule for it.

One instance of NP-internal case marking, one of the animacy effects listed by Dahl and Fruenda (1996), has been studied for Low Saxon. Low Saxon is a Germanic language spoken in northern Germany and Netherlands, and it is closely related to Dutch. Strunk (2004) has performed a corpus study of various possessive constructions in this language to check for animacy effects (using again another hierarchy, HUMAN - ANIMATE - ORGANIZATION - CONCRETE - ABSTRACT). The author collected samples of four possessive constructions, and found that, when the possessor is low on the animacy hierarchy, the Prepositional Possessive construction is often used. When the possessor is more animate, the three other constructions are chosen more often, and this effect cannot be reduced to other factors. He also notes that this choice is similar to English’s choice of using ’s possessives or the of-possessive, with the of-possessive corresponding to the prepositional possessive of Low Saxon that was found to be used more with inanimate possessors. Similar constructions occur in Dutch, so these findings may have implications for Dutch as well. Either way, it is another case of an animacy effect where there is not a hard rule, just a tendency, but the tendency was observed with a statistical (logistic regression) model.

A study by Mak et al. (2002) argued that animacy also affects the processing of relative clauses in Dutch. A common finding in psycholinguistic literature has been that subject relative clauses are easier to process than object relative clauses in various languages. Object relative clauses take longer to read for this reason. In subject relative clauses, the relativized element has the subject function in the relative clause:

\begin{align*}
(13) & \textbf{The cat} that touched the apple fell off the table. \\
\end{align*}

While in object relative clauses, it has the object function in the relative clause:

\begin{align*}
(14) & \textbf{The apple} that the cat touched fell off the table. \\
\end{align*}

Reading times for subject relatives such as in example 13 are usually found to be shorter, indicating less processing difficulty.

However, Mak et al. (2002) has found that this only holds for object relative clauses when the object is animate, in a Dutch language study. They used Dutch because object and subject relatives are disambiguated later than in other languages (only once the auxiliary verb is reached in the verb cluster located at the end of the clause). The constructions are also more similar to each other in Dutch, excluding the influence of other factors. They use the following example sentences (I have highlighted the object):

\begin{align*}
(13) & \textbf{The cat} that touched the apple fell off the table. \\
(14) & \textbf{The apple} that the cat touched fell off the table. \\
\end{align*}
For object relative clauses with inanimate objects, such as example 14, reading times were similar to those of subject relative clauses (13), negating the usual difference in processing difficulty. It is theorized that readers interpret the animate noun phrase (NP) as the subject, when the two NPs involved in a relative clause differ in animacy. This disambiguates them at an earlier stage than relative clauses involving two inanimate NPs, somehow preventing the processing difficulties for object relative clauses from occurring. This finding indicates that animacy at least plays some role in human sentence processing, possibly guiding the choice of whether a clause should be read as an object or subject relative. This also supports the idea that knowledge of animacy categories may be beneficial for sentence parsing in Dutch.

Another animacy effect that has been studied for Dutch is the use of non-canonical word orders, specifically object fronting (Bouma, 2008). The hypothesis is that, with the more common role assignment of an animate subject or an inanimate object, speakers are more likely to use object fronting. If object fronting is used when the roles are reversed, it would become more difficult to figure out the correct roles, and the message would become less clear. Therefore, object fronting is predicted to be more common when the subject is higher on the animacy hierarchy than the object. The author confirms this tendency by examining a corpus of spoken Dutch and fitting a logistic regression model to it, though with the reservation that there are few instances where the object is higher on the animacy hierarchy in the data he annotated. However, since there are some results from other languages along the same lines, they are likely to be true for Dutch as well.

The reasoning that was used here is similar to hypotheses in differential object marking, as in the Russian example discussed earlier. It has been found across languages that this marking occurs more often when a noun has an unexpected role. For example, direct objects are most commonly inanimate, and
languages have a tendency to case-mark only animate nouns in that slot (accusative case marking). The same effect was observed with subject nouns, which are most often animate — languages may mark inanimate subject nouns (ergative case marking). This phenomenon is called differential case marking, and is one of the most widely studied animacy effects. This kind of cross-linguistic typology research is one other way of studying tendencies. By looking at a large variety of languages, it is possible to observe general tendencies in grammar, even when categories and rules are used, since they differ for each language. A survey of differential case marking in a sample of 200 languages backs up this observation about unexpectedness and shows that it is a tendency across languages (Fauconnier, 2011). Unexpectedness is seen as the main cause — there are many languages where inanimates cannot be the agent of a transitive clause at all, and when it does happen, it is marked.

Such observations may also provide an indication for processing effects of animacy. If a language does not mark animacy in a specific situation, but other languages often do, this may indicate that there is still some animacy effect to be found, even though it is not explicitly marked.

These processing effects indicate that animacy is involved in language comprehension and production even when it is not explicit in the grammar. Therefore, animacy may also be a relevant factor in the domain of automated natural language processing, where it has often been ignored. In the next section, I will discuss why this might be.

2.3 Animacy in natural language processing

Even though the animacy hierarchy plays a role in various parts of linguistics, it has mostly been overlooked in natural language processing. Zaenen et al. (2004) theorize that this is because animacy, in English, often doesn’t influence grammaticality, although it is important for felicity. Because English is the language with the most language resources and corpora, and the language for which the majority of NLP tools are developed, its properties have a strong influence on the design of corpora, annotation schemes or NLP algorithms. Many NLP applications are mainly interested in grammaticality, for which animacy is not an important distinction in English. However, felicity aspects, which concern the acceptability of sentences, can be important as well, particularly in natural language generation tasks. And the processing effects and probabilistic tendencies discussed in the previous section may also be relevant for statistical models of language. As the linguistic literature shows, incorporating animacy into NLP tools may prove to be more useful for other languages.

There have not been many attempts to automatically acquire animacy information of words. One method of lexical acquisition to obtain animacy information through WordNet has been used by Orasan and Evans (2001). However, this only works for languages that have a large lexical-semantic database. They used the semantic hierarchy of the database to infer that all the words that are hyponyms of the high-level (generic) semantic concept of an animate entity, should be considered animate. Section 3.2 discusses this in more detail.
The other main approach for obtaining animacy information is to classify nouns based on some of their properties or features, extracted from a corpus, that might indicate their animacy status. In the next chapter on related work, I will discuss these classification efforts.

In the previous sections we have seen many animacy hierarchies and categories, all useful in different situations. Natural language processing tasks also work with classes and require or produce annotation, which raises the question which hierarchy should be used. As noted before, animacy can be viewed as a grammatical category and this categorization doesn't always match the semantic view of animancy. Therefore we can say that for most computational linguistics applications, simply using biological distinctions is not sufficient. We want to model the way animacy is used in language, which may not match the biological reality (similar to how grammatical gender doesn't always match biological gender). There is no objective measure for animacy, it's based on the way groups of speakers interpret these nouns (Zaenen et al., 2004).

For the animacy classification task, a broad classification using the main categories is still possible with a good degree of certainty, even though there is a lot of uncertainty about the middle area of the animacy spectrum. Whether more specific categorization is needed depends on various factors, such as the availability of training data for the desired categorization, the NLP task that the animacy-annotated data is to be used for, or the annotation scheme itself. In the next section, I will discuss two such schemes that have been used for corpus annotation. Even though they are not specifically designed for NLP tasks, one may argue that corpus annotation is a goal of animacy classification, and they are therefore a good starting point.

2.4 Annotation schemes

In many cases, a basic inanimate-animate-human distinction is not fine-grained enough, and more distinctions have been found in language. In order to perform research with a more detailed animacy hierarchy, a more detailed annotation scheme would be needed. Here, I will discuss two of them.

2.4.1 Referentiebestand Nederlands (RBN) format

The Cornetto lexical-semantic database for Dutch uses the annotation scheme of the RBN dictionary (Martin et al., 2005), which is one of the two data sources (the other being Dutch WordNet) that was used to create it. For animacy, this annotation scheme uses a hierarchy with animate and inanimate categories at the top level, as well as the institution category. The animate and inanimate categories are subdivided further. The scheme, as shown in figure 2.1, is limited in scope but designed with annotation in mind - for example, there are categories like concrother to handle borderline or exceptional cases. Since this hierarchy was designed for a lexical-semantic database, it was likely designed with a semantic notion of animacy in mind, rather than a set of grammatical classes.
The **institution** category is classified as neither animate nor inanimate, because while they are not animate, they can perform as an agent in a sentence. Nouns like 'government' belong to this class, or even names like 'Belgium', referring to the Belgian government, even though it would normally count as a **place**.

The animate category is subdivided into **human** and **nonhuman**. The **nonhuman** category is used for nouns that refer to plants or animals. However, nouns referring to parts or products of animate entities, such as body parts or fruits, are classified as **concrete** inanimate nouns.

For inanimate nouns, a distinction is made between **place**, **time** and **measure** nouns. Apart from the classes mentioned thus far, all other nouns are classified as either **concrete** or **abstract**. Concrete nouns can be substances, such as 'water', or artefacts, human-made objects such as 'cookie'. Artefacts are always count nouns, though substances may not be. This leaves some nouns that belong to neither category, such as 'orange' or body part nouns, they are classified as **concr other**. The **abstract** category is also subdivided. **Dynamic** nouns presuppose some event or point in time, for example 'presentation', while **nondynamic** nouns, such as 'hate' or 'attribute', do not.

This annotation scheme was designed for use in a lexical-semantic database, and its animacy hierarchy is meant for labeling the nouns in such a database with the semantic property of animacy. In other words, it can be seen as dictionary information. It was not used to label text corpora, in which each token of a text would be labeled with its animacy property. In that case, some instances of 'Belgium' in a text might be labeled as **place** when referring to the country, and others as **institution** when referring to the government. That would be a token-based annotation. Lexical-semantic databases like Cornetto are based on word senses. Each sense in the database is its own entry, with an animacy category, among other kinds of labeling. A noun can have multiple senses, each with its own animacy category label. As far as we are aware, token-based annotation for animacy in Dutch hasn’t been done directly. However, a recent corpus annotation project called DutchSemCor is working on token-based word sense annotation (Vossen et al., 2012). In this corpus, each word is annotated.
with its sense as listed in the Cornetto lexical-semantic database. These senses have associated animacy information, if they have been annotated, so in some way this can be considered to be token-based animacy annotation.

2.4.2 Stanford-Edinburgh paraphrase project format

Zaenen et al. (2004) describe an annotation scheme for animacy in English language corpora, developed for a project by O'Connor et al. (2004) and also used for a Stanford-Edinburgh collaborative project on paraphrasing (Bresnan et al., 2002). This scheme uses three main categories — human, other animates and inanimates, the latter two being subdivided further, though not into a full hierarchy. The subcategories are:

- Other animates: organizations, animals, intelligent machines and vehicles.
- Inanimates: concrete inanimate, non-concrete inanimate, place and time

In this section, I will discuss the differences with the Cornetto format. Even though the formats have been developed for different languages, they seem to be based on a semantic notion of animacy rather than a grammatical one. The semantics of animacy are unlikely to be very different in languages as related as Dutch and English, and so the annotation schemes should be comparable.

In the top-level categories we can see that this scheme already makes a distinction between humans and other animates, not grouping them all into animates, though such a category could easily be made by merging the two. Furthermore, the organizations class, which more or less matches institution from the other scheme, is categorized under other animates rather than being a top-level category by itself. It appears that the designers of this scheme disagree with the RBN scheme that organizations are not animate. However, organizations can perform as an agent in a sentence, much like animates, so they may act more like animate entities in texts. An automatic classifier would probably have less difficulty classifying them as such, according to this scheme. In general, this scheme seems to have a broader notion of animacy than the RBN one, also including intelligent machines and vehicles, and unlike RBN, plant life is excluded from animacy and classified as concrete inanimate nouns instead.

These choices indicate a definition of animacy that is based on what entities can act as agents, rather than being based on biology. Other differences between the two schemes are minor, with the RBN scheme having some additional categories and subdivisions at the lowest level.

The reliability of some animacy annotation schemes has also been investigated. In an experiment about the reliability of a complex scheme, it was found that the reliability of human annotation was good for the major, outer categories (human and non-concrete inanimate), with a kappa value of K=.92 (more than 0.8 is considered good) but not for the intermediate categories, which was thought to be caused by too informal definitions in the investigated scheme. The annotators did not have the same intuitive understanding of the categories as the developers. In particular, there was confusion between human and organization. In addition, rare categories such as vehicles were
sometimes forgotten about by the coders, and vague pronomial references were resolved differently by different coders, giving them different animacy categories (Zaenen et al., 2004).
Chapter 3

Related work

In this chapter, I will discuss some work related to automatic animacy classification. There have been several previous attempts at animacy classification, though none for Dutch. First, I will summarize an article that called for an increased interest in animacy for NLP tasks. In section 3.1, I will discuss various articles about an animacy classifier for Norwegian and Swedish, based on morphosyntactic distributional information extracted from corpora. Section 3.2 discusses work that uses a large lexical-semantic database, WordNet, for animacy classification. Section 3.3 discusses a similar method based on an electronic dictionary. In section 3.4 a method based on web N-grams will be discussed, requiring a lot of data but not much annotation, and lastly, in section 3.5, a classifier using lexical distributional information extracted from corpora will be discussed.

Seeing that animacy was often ignored in natural language processing, Zae- nen et al. (2004) discussed two ongoing animacy annotation projects, and discussed the importance of using this data in computational linguistics. They focus on felicity aspects of animacy and the ‘accessibility hierarchy’ as the motivation for this. Accessibility scales are theorized to influence the grammatical prominence of entities in the discourse - for example, whether they are fronted or relativized or in what thematic role they are realized. Animacy is an example of such a scale, along with person (which is sometimes put on one scale with animacy) and definiteness. They state that these scales are known to play an important role in the organization of sentence syntax and discourse in linguistics, which we have also seen in the previous chapter, and that they are not widely recognized in computational linguistics.

They go on to list some examples of animacy and grammaticality, as we have seen in section 2.1, and conclude that these phenomena mostly occur in morphologically richer languages than English. For English, they argue that animacy is relevant for felicity aspects, since it affects choice of syntactic constructions, similar to some probabilistic examples we discussed in section 2.2. This, they argue, makes animacy an important factor for natural language generation as well as automatic translation, in which choices between various grammatical ways of expressing the same thing need to be made. In the same way that choice
between constructions is probabilistic and dependent on some factors such as animacy in natural language, a natural language generator could assign weights to different constructions depending on the animacy of the entities that need to be realized in the discourse — for example, inanimate entities are more likely to be the object.

While this article does not discuss classification directly, there is a detailed discussion of a manual corpus annotation effort, of which we discussed the annotation scheme in section 2.4. This is partially relevant to automatic classification as well - animacy categories that are confusing for human annotators, may also be more difficult to classify automatically. The fact that resources are being invested into manual animacy annotation, and that the resulting corpus is being used, also indicates that there is a need for more efficient and larger-scale annotation options, such as those provided by automatic classification.

We will now proceed to discuss some existing animacy classifiers. These classifiers were all developed for different languages, with different resources available, and different definitions of animacy. It is therefore difficult to compare them directly, however, it is interesting to look at the different methodologies and approaches to the task and the motivation for the choices that were made.

3.1 Animacy classification based on morphosyntactic corpus frequencies

There has been a large project on animacy classification for Norwegian and Swedish by Lilja Övrelid, who has published various articles on the topic over the years. A first version of the classifier was described in Övrelid (2005). It is a decision tree classifier for Norwegian nouns that is based on syntactic and morphological distributional features, which are extracted from a dependency-parsed corpus.

The idea of using such features was taken from an earlier verb classifier, which classified on relative frequency data for each verb in a certain class, meaning that the features of every instance (token) of a verb were counted, adding them up for the lemma (type). This type of feature is therefore not context-sensitive; it adds up information of all instances of the verb, and cannot classify individual instances. However, since features of multiple instances are extracted, there is more information to base classification decisions on. Övrelid (2005) adapted this idea for animacy classification, using some linguistically motivated features that can be counted in this way.

**Subjects and objects** As noted before, subjects are typically animate and objects are typically inanimate. This is not a hard rule, but nevertheless such tendencies can be used in probabilistic systems such as classifiers, and therefore the frequency with which a noun occurs as a subject or an object can be used as a feature here. The authors claim that in simple Norwegian transitive sentences, around 70% of the subjects are animate, and 90% of the objects are inanimate (Övrelid, 2004).
**Passive** The semantic role of agent is also strongly associated with animacy — agents are normally not inanimate, with some exceptions such as organizations. But because semantic role information isn’t easily available in corpora, the author approximates it with the passive construction. Transitive constructions are passivized more frequently if the demoted subject is high on the thematic role hierarchy, for example an agent:

(18) The ball was kicked by the girl (AGENT)

Nouns that are used as demoted subjects are quite likely to be agentive, and therefore likely to be animate, and demoted subject relative frequencies seem to be a useful feature for animacy classification.

**Anaphoric reference** Personal pronouns in Norwegian, as well as English, encode the animacy of their referent (animate *he/she* and inanimate *it*). While this is useful information, to know this one must find out what the pronoun refers to (coreference resolution), which is a challenging NLP task in itself and actually also a task where having animacy information would help. The author solved this by using a simple approximation of anaphoric reference. Personal pronouns are more likely to refer to entities that are more salient and more recent. The clearest case of this is described by the author:

If a sentence only contains one core argument (i.e. an intransitive subject) and it is followed by a sentence initiated by a personal pronoun, it seems reasonable to assume that that these are coreferent.

An English example of this situation is the following:

(19) The man laughed. He couldn’t believe it.

For the anaphoric reference feature, only instances of this simple case of anaphoric reference are counted, comparing whether an animate or inanimate personal pronoun is used. The plural *they* is ambiguous for animacy in Norwegian as well as in English, but because *they* (in English) was found to refer to animate referents in 76% of cases, she assumed it to be similar in Norwegian and included it.

**Reflexive** Reflexive pronouns can also indicate animacy in a more indirect way. Their advantage is that they can be resolved locally which is trivial to do automatically:

(20) The teacher hurt himself.

In Norwegian, reflexive pronouns do not use the animacy dimension. Instead, since reflexives are mostly agentive, the fact that the noun was used reflexively is used as a feature that indicates animacy, in the same way that the passive feature is used.
Genitive -s Norwegian has a genitive case which typically, though not always, indicates possession. This is the only case marking in the Norwegian language, and similar to English or Dutch, animacy is not marked. Possession often involves an animate possessor, though not always (part-whole relationships are an exception). It can therefore be a feature for the animacy classifier, especially since it is marked in Norwegian, making extraction of this feature easy.

These distributional features were automatically extracted from an annotated corpus. The corpus is annotated with underspecified dependency trees, which provide enough information for automatically obtaining the features described above. After feature extraction, a noun is thus represented by a feature vector where the values are relative frequencies over every instance of the noun. Since the features are all related to animacy in some way, this provides the necessary information for animacy classification. The features indeed had quite distinct averages for the animate and inanimate classes, although some features were found to be quite sparse, such as usage with the reflexive, which only occurred 558 times in a 15 million word corpus. Such features are likely to be effective only for high-frequency nouns, and the classifier indeed turns out to be more accurate on such nouns.

She uses a simple two-way animate-inanimate distinction for classification, using only 40 nouns as training data. A weighted decision tree is used, in which each node in the tree represents a decision point, where a branch is chosen based on the noun's properties. Each leaf of the tree is assigned either the animate or inanimate class, representing the classification outcome.

The classifier is evaluated using 10-fold cross validation, reaching a classification accuracy of 90% when all features are used, although the set of nouns is quite small, and only high-frequency nouns were used, which makes the evaluation unrealistically easy. The nouns all occurred more than 1000 times in the 15 million word corpus. This classifier was also tested on nouns occurring around 100 times in the corpus, which reduced the accuracy to 65% with all features, although backing off to only the more frequent features raised the accuracy again. This backing off idea was explored further in a followup article (Øvrelid, 2006). Her explanation for the sparseness issues is that most of the features (i.e. reflexivity) indicate animacy rather than inanimacy, and when less information about these features becomes available, animate noun feature vectors start to become more similar to inanimate noun feature vectors. The inclusion of some features that specifically target inanimacy could be a solution here, though this option is not explored by the author.

With regards to the backing-off, it was found that the most high-frequency features, relative object frequency and relative subject frequency, performed best on lower-frequency nouns (50-100 occurrences) providing classification with near 90% accuracy as well. One other backing-off option was explored — the use of a classifier specifically trained on nouns of a similar frequency, which resulted in slight improvements in some low-frequency cases but nothing dramatic.

Next, the author applied the same technique to Swedish data, and used the resulting animacy information to improve a Swedish dependency parser (Øvrelid, 2008). This time, a different algorithm was used - instead of decision trees, she used the k-nearest neighbour algorithm as implemented in TiMBL.
the Tilburg Memory-Based Learner, though the article does not elaborate on this. The feature set was also expanded with the following additional features:

**Syntactic features** In addition to the previously mentioned subject and object features, involvement in other dependency relations is also included, though this is not motivated. They include all dependency relations that nouns may occur in, such as root, conjunct, determiner, and prepositional complement.

**Morphological features** In addition to case, every morphological distinction for nouns in Swedish is now included: gender, number, definiteness, date and quantifying noun.

**Proper nouns** She used named entity recognition (NER) as an additional source of animacy information. In NER, proper nouns are categorized into semantic categories like place or person. An automatic NER system for Swedish was used on the data, and nouns that were tagged person are likely to be animate entities, making this category a useful feature. While it seems strange to use a NER system to obtain animacy features (animacy information can be used in NER!) it would be difficult to gain information on low-frequency proper nouns in any other way, since they would rarely occur in the corpus.

High accuracy scores are obtained with this setup, although the Swedish data set is largely inanimate and therefore the baseline is high. It was also found that adding the animacy information, which was automatically obtained using the classifier, improved the dependency parser accuracy. This shows that animacy information, even when automatically obtained, may indeed help other NLP tasks.

Lastly, Øverlid (2009) provides a more detailed evaluation of an improved version of the Swedish classifier, addressing some issues such as the type-based nature of the classifier and the granularity of animacy categorizations.

The difference between two levels of animacy annotation are discussed, type-level or token-level. The classifier provides type-level annotation — it can output an animacy category for each noun type (lemma), using feature information about every instance of it in the training data. The alternative would be token-level annotation — examining each instance of the lemma and assigning a category to it. This task is more difficult, since it is context-sensitive, and less feature information is available (only that of a single instance). The author finds some cases where a type-level approach is often insufficient to accurately annotate animacy in all contexts. These are abstract nouns (such as quantifying nouns) and nouns used in different contexts that shift their reference (such as idioms).

The article also provides various evaluation results of the system. The system reaches 96.8% accuracy for nouns with a frequency of more than 100 in their corpus (1668 instances). The baseline score for this task, a score that would be obtained by classifying every noun as inanimate, is 90.5%. They compare their k-nearest neighbour classification to another common machine learning
algorithm, support vector machines, but there is no significant difference. They also show that the (less common) animate class is more difficult to classify, particularly for lower-frequency nouns.

Furthermore, they also demonstrate that the task of dependency parsing may benefit from animacy information by taking a standard language-independent dependency parser and training it on a treebank with and without automatic animacy annotation. The parser that was trained on the animacy-annotated achieved a significantly higher labeled attachment score. An error analysis shows that the improvements are mostly in the labelling of object and subject relations, and subject predicatives.

3.2 Animacy identification using lexical-semantic databases

Orasan and Evans (2007) approach the task of animacy identification from the viewpoint of anaphora resolution. They plan to use animacy information for improved anaphora resolution. They filter out any candidate referents that do not agree in animacy with the pronoun — for example, *it* cannot be used to refer back to *the man*. Their methodology is based on this, and the anaphora resolution approach shows in their definition of animate NPs, which they consider to be any noun that is referred to using *he, she* or related animate pronouns. This contrasts with all of the previous discussion, where animacy categories based on semantics were used. This also means that some entities that are often fairly high on the animacy scale, such as *baby* or *family*, are considered inanimate.

They present two methods, which are both based on WordNet, a large lexical-semantic database for English. WordNet is organized hierarchically through hypernym and hyponym relations between word senses, also called synsets (synonym sets). This brings up the resource restriction that such a database needs to be available for the methods to work, as they both rely on such a hierarchy. The main advantage is that all word senses can be taken into account when making a decision about animacy, though one can argue that Øvrelid’s method also does this implicitly, as long as the senses are used in the corpus. Their two methods are the following:

**Rule-based method**  At the top of the WordNet hierarchy, there is a small set of generic words known as *unique beginners*. Some of them are related to animacy and large parts of their hyponyms are animate. This information can be used to infer the animacy of anything occurring under these unique begin-

ners in the hierarchy. There are three animate noun beginners, *animal, person* and *relation*. In addition to that, some verb-category beginners are identified of which the verbs in their sense hierarchy should have animate subject NPs, namely *cognition, communication, emotion* and *social*. In cases where a noun has multiple senses, some of which are inferred to be animate and others inanimate, the ratio of animate to inanimate senses is computed and a threshold can be set to classify them as either animate or inanimate.
**Machine learning** This method was developed later to improve upon some weaknesses of the previous one. Since the animacy of the unique beginners is not certain, they now use an annotated corpus to identify the animacy of synsets. This method is even more specific, requiring a corpus that has been annotated with WordNet senses as well as with animacy information. Instead of propagating animacy top-down from unique beginners, they now propagate the information bottom-up, starting from unambiguous terminal nodes, for which each occurrence in the corpus was assigned to the same animacy class. They use a statistical method for this upwards propagation, since unambiguous terminal nodes are apparently rare and it is a difficult task to decide on the animacy of more general nodes. They also leave the option open of assigning neither class, when a node is too ambiguous. This decision is made using the chi-squared test, comparing the population of senses that were annotated as animate to a situation in which every sense would be animate, and testing for the significance of this. This approach is used to classify all the noun senses (including an ‘undecided’ class), as well as the verb senses for their subjects, resulting in an animacy-annotated WordNet, which is then used for classification using machine-learning.

They also used TiMBL’s k-nearest neighbour classification as their algorithm, like Øvrelid in her later articles. Unlike Øvrelid, they used the following features:

1. The number of animate and inanimate senses of the word (as inferred using the previously discussed methods).
2. For the heads of subject NPs, the number of animate/inanimate senses of its verb.
3. The ratio of the number of animate singular pronouns (e.g. he or she) to inanimate singular pronouns (e.g. it) in the whole text.

So, they use animate senses as a feature for animacy. It is a direct way of transferring the inferred WordNet information to a corpus, making the classification task relatively easy — in this case the difficulty is in obtaining the features. They also take a lower-resolution approach to pronoun resolution than Øvrelid.

Their approach is still largely type-based, although they take some steps to make it as sensitive to their data as possible. They test a word-sense disambiguation technique to weigh the importance of each sense, and therefore how much it should contribute to the animacy decision, by counting the relative frequency of the sense compared to other senses of the word in the text. This is based on the intuition that more prominent senses should count more towards the decision of identifying a word as animate or inanimate. However, the system performs better without this step. The classifier was tested on a different corpus than the one it was developed on, in which the baseline accuracy (classifying everything as inanimate) is 88.2%. The classifier accuracy (with the machine learning method) is 97.7%. They perform an additional evaluation by applying the classifier to the task of anaphora resolution, using a similar setup as the parsing evaluation of Øvrelid (2009). An existing anaphora resolution is used, one instance is trained on standard data and another on the automatically animacy-annotated data. They observe improvements in accuracy, though
the evaluation is poorly set up since the anaphora resolver is not suited to the domain, and performs poorly in all cases.

3.3 Semi-automatic labeling using a dictionary

de Ilarraza et al. (2002) describe an animacy annotation effort for the Basque language. They needed information about noun animacy to solve some common ambiguities in machine translation to Basque. However, Basque is a fairly under-resourced language, so no lexical-semantic database was available at the time. They instead used the semantic relationships described in an electronic monolingual dictionary to classify a large number of words, starting from a small, manually annotated seed set of 100 nouns.

The idea is similar to that of Orasan and Evans (2007), and they used synonymy relations in addition to hypernyms and hyponyms. They first annotated 100 nouns manually. The interesting difference is the resource — a dictionary designed for human use rather than computational purposes — and the method of extraction. Hypernymy and hyponymy can be inferred from definitions such as:

\[(21) \text{aeroplane. vehicle that can fly}\]

The 100 hypernyms that occurred most frequently in such definitions were annotated, and their hyponyms as well as synonyms were extracted. These were then assumed to belong to the same category as the hypernym. This process was repeated iteratively. There was also a reliability measure similar to that of Orasan and Evans (2007), and a class for ambiguous nouns.

The method achieves over 99% accuracy with a coverage of 68.2% in the classification of all common nouns in a 1 million word corpus. There is no way for this method to handle unknown words — only nouns that occur in the dictionary and that are linked to other nouns can be covered.

3.4 Animacy knowledge discovery from web-scale N-grams

Ji and Lin (2009) note that animacy, as well as gender, is a strong factor for predicting person mentions in mention detection. Mention detection is the task of detecting references to entities, i.e. persons. Motivated by that task, they built a system for animacy knowledge discovery from Google N-grams Version II, a very large corpus of n-grams gathered from the web, automatically annotated with part-of-speech tags (but not full syntactic trees). They use a simple lexical pattern, based on one of the (few) cases in which animacy affects grammar in English, to obtain animacy information. The pattern exploits the fact that relative pronouns express animacy, and can refer to nouns that occur in the main clause directly before them:
He met the **writer who** wrote the new book.

She saw the **place where** she had dinner yesterday.

The relative pronoun occurs either directly after the noun in the sequence of words, or there is a comma in between, but nothing else. No syntactic knowledge is required to extract this pattern. Since this only works for one very specific pattern, a lot of data is needed to make this work, but Google N-grams provides this — this pattern occurred 664,673 times.

This grammatical phenomenon was discussed in section 2.1. Unfortunately, it would not work for Dutch. This method uses a language-specific pattern that detects this English grammatical phenomenon, this construction does not work this way in Dutch and does not have an animacy-based distinction. There is another possible construction that would work for Dutch, also discussed in section 2.1, but it is far less frequent. The Dutch version of Google N-grams isn’t as big, either.

The authors used the animacy information obtained from this pattern (as well as gender information obtained with similar patterns) in an unsupervised mention detection system. The animacy information had the largest impact on their system’s performance, indicating its importance for this task.

### 3.5 Animacy classification by sparse logistic regression

Baker and Brew (2010) describe an approach that they claim is multilingual, tested on English and Japanese. They take a two-category approach to animacy (**+ANIMATE** and **-ANIMATE**) and use a Japanese data set that is partly manually labeled, meaning that some nouns are lacking labels. Their focus is on classifying the less frequent items, which was already shown to be more difficult by Øvrelid (2005) than classifying frequent items. As their classification features, they use frequency counts of verb-argument relations (distributional lexical features), rather than the morphosyntactic features used by Øvrelid. They also try some additional techniques, such as using English animacy classification to classify English loanwords in the Japanese data.

Japanese has several instances of explicit animacy marking, which makes it fairly easy to automatically label at least the nouns that occur in such constructions in a corpus. They also used English resources to label English loanwords in Japanese. They then use these nouns as a seed set for training a classifier, with the following features:

**Subject (Object) Frequency**  The frequency with which a noun occurs as a subject (or object) of specific verbs. This means that each verb is a feature, with the values representing the occurrence of the noun and the verb in a subject relation. This can be contrasted with the subjects and objects feature used by Øvrelid. They only counted the number of subject relations in general, while this work counts the number of subject relations for each verb. This feature is
motivated by the fact that verbs often have semantic selection restrictions that can involve animacy — for example, a subject of the verb *to think* is normally sentient, and therefore animate.

**Verb Animacy Ratio** The number of animate subjects for this verb, divided by the total number of subjects, as found in the training data. This was used to replace raw frequency counts for the subject/object frequency features.

**Average Verb Animacy Ratio** The average animacy ratios of the verbs that occurred with a noun at least once. This is a separate feature that is generalizing over the other ones. It was counted once for each noun.

They use a Bayesian logistic regression classifier, for its ability to handle a large number of features. Taking each verb-subject and verb-object relation as a feature makes this an issue. With this setup, they run three experiments, trying various additional methods:

**Baseline** Japanese animacy classification using the features described above. Much like Orasan and Evans (2007), they find worse performance for animate nouns than for inanimate ones. They obtain accuracies up to 88% using the average verb animacy ratio, their best-performing type of feature value, though covering only 36% of the data set.

**Equivalence classes** In this experiment, nouns were grouped into equivalence classes prior to classification. For example, all nouns ending in *-man* were considered kinds of men, though they used Japanese data, and in Japanese these suffixes are more homogenous than in English, often consisting of single characters. This is particularly the case for words of Chinese origin, which tend to have a compound structure. For example, there is a specific (single character) suffix for ‘person’, *-jin*. The classes were then formed by grouping all the items ending in the same kanji character, even though this is not 100% accurate. One advantage was the reduction in size of the feature vectors, which are normally quite large when lexical features are used. This approach leads to accuracy scores of 95% and a larger coverage (51% of the data), this time with the verb animacy ratio feature set.

**English loanwords** In their test data, the authors encountered many English loanwords that did not occur in the training data. It was their biggest class of unknown words, and the use of Japanese suffixes does not help to classify them. They attempt to solve this problem by making an English classifier and transliterating the data. They used only transitive subject relations as features. Then, they automatically translated the English loanwords from Japanese to English, and grouped multiword compounds. On this English data, they reach 88% accuracy with plain subject counts — the verb animacy ratio features did not appear to work for English. The ratios did not differ enough between nouns, implying that English verbs are not very sensitive to their subject’s animacy. Applying the English labels to the Japanese loanwords yielded an accuracy of
87.9% for Japanese with 97% coverage. The remaining 3% did not occur in any object or subject relation in the data set.

Even though the method is multilingual, it performs better on Japanese than on English. It relies on some features that are more associated with animacy in Japanese than in English, and features that are easier to extract for Japanese.
Chapter 4

Data and methodology

The task of animacy classification can be summarized as identifying to which animacy class a noun belongs. As we have seen in the previous chapter, which discussed several examples of animacy classifiers, there are various approaches to this task, including differences in the way the task is defined. Considering the results of the Swedish animacy classifier, it is logical to use a similar methodology. However, differences between the languages and the available data should be taken into account. In this section I will discuss how the problem was approached for Dutch animacy considering the available data.

4.1 Classification task

In a classification problem, it is important to define the classes well. Classes need to be clearly distinct from one another, at least to humans, so that they have distinct properties that can be used to differentiate between them. A good example of this are the top-level animacy classes of the Stanford-Edinburgh paraphrase project discussed in section 2.4.2. In this three-way division into humans, other animates and inanimates, the 'other animates' class was defined to include any nouns exhibiting animate properties, i.e. including vehicles and intelligent machines, rather than just following biological animacy. It could be argued that these entities are likely to be more like animates linguistically. An automatic classifier, using context features, would be able to detect this.

In our choice of classification scheme, we are constrained by the annotation scheme of the Cornetto lexical-semantic database (discussed in section 2.4.1), because it is the only large-scale resource containing animacy information for Dutch, and it will be used as gold-standard and training data. The next section will elaborate on this. The scheme includes a large hierarchy for inanimate entities, though we are more interested in animate-inanimate distinctions. For this, the scheme can be generalized to a three-way classification of HUMAN, NON-HUMAN, and INANIMATE. That includes both of the animate categories, and the inanimate category as a single one. we have included the (small) Institution category in NONHUMAN, because it seemed to contain other institutions.
Unfortunately, there is no further subdivision of the nonhuman animate class available. This greatly limits the options for defining the scope of these classes in the classification task. The only choice is to follow the classes used by the existing annotation scheme. In the next section, we will discuss the data in these classes.

Apart from the classification scheme, there are several levels of detail at which animacy classification can take place. One is type- or lemma-based classification, in which an animacy class is assigned to each word. Another is word sense based, in which each sense of a word receives an animacy class. This is only possible when word senses are known, which is the case in a lexical-semantic database, for example. Seemingly the most challenging method is token-based classification, in which each instance of a word in a text receives an animacy class, which may depend on their specific sense and context.

In the type-based method, where animacy is determined for each type, the result of classification is a sort of dictionary of nouns occurring in the data and their animacy class. This necessarily means that any and all senses of a lemma get the same classification, which may not always be accurate - there could exist nouns with both an animate and an inanimate sense, i.e. ‘contact’ which as a noun can both refer to a human (someone to have contact with) or an action (making contact with something). However, it does not require any information about word senses or a word sense disambiguation system. The animacy property resulting from such classification is context-independent (Øvrelid, 2009).

Type-based animacy classification can make use of any information associated with the lemma, for example, context information obtained from corpora. Features that may distinguish animacy are extracted from an annotated corpus for each occurrence of the lemma. A classifier can then be trained on those features. An example of a feature would be how frequently the noun occurs in a subject rather than an object position in the whole corpus. The Swedish animacy classifier (Øvrelid, 2009) uses a type-based approach, because only a small amount of nouns were found to have varying animacy depending on context, so the context-independency didn’t prove to be a large problem. An examination of SenCor, a Dutch corpus with word-sense level annotation, showed that, of the 2072 nouns annotated with a semantic type, only 34 types (1.5%) are ambiguous in terms of animacy. This indicates that Dutch texts may also have low animacy ambiguity, and that it may not be very important to perform the more difficult tasks of word sense level or token-based animacy classification.

In the token-based method, the classifier operates on each instance of a noun in an annotated corpus and determines its animacy based on distinguishing features that the token has. While type-based classifiers have access to all the different contexts the noun appears in, a token-based classifier will only have access to one set of context information and therefore has less information to base a decision on. Sometimes the meaning of a word depends on the context, such as with homographs, and a type-based classifier would not take this into account. A token-based classifier would be able to avoid the type of errors stemming from ambiguous nouns. Another argument for a token-based approach is that animacy is about the animacy of the referent, and the referent can be different for each token.
For the Dutch language, there is no corpus with animacy annotation to use as gold standard data, so the token-based approach is not an option for training. Instead, the Cornetto lexical-segmental database was used to obtain a dictionary of nouns and their animacy status. This process will be described in the next section. The DutchSemCor corpus (Vossen et al., 2012) with word sense annotation that is linked to Cornetto senses could potentially be used for a token-based approach, however, it has only been made available very recently and we were not able to try this.

As we have seen in the related work chapter, some kind of linguistic context features are used to make a classification decision. To obtain this information, we need to find some instances of the nouns in an annotated corpus, in which we can find, for example, how many times the noun occurs as an object or a subject. This is only possible if the corpus includes semantic role labelling annotation. A lexical-segmental database like Cornetto does not provide any such context. So, the dictionary of nouns was looked up in the Lassy Large corpus, a large corpus of syntactically annotated Dutch sentences containing around 1.5 billion words, to retrieve relevant contextual information that can help in classification. This process will be discussed in section 4.3, and the extracted context features will be described in chapter 5.

Classification tasks generally involve a set of data for which the categories are known. This 'gold standard' data is used to train a classifier to make correct classification decisions. Since we will use a statistical method, it will be beneficial to have a training data set that is as large as possible, containing a large amount of examples for each class. Gold standard data sets can be created by having humans manually annotate the data, but it is difficult and costly to obtain large data sets in this way. It is better to use an existing set. For Dutch animacy, this would be the Cornetto lexical-segmental database.

For evaluation purposes, a part of the gold standard data should also be set aside. A list of words with a known animacy category that was not involved in the training can be used to simulate classifying unknown words, which is necessary for a fair evaluation. After all, if the classifier was already trained on a word, it is trivial to classify it. More advanced evaluation techniques perform multiple rounds of evaluation in which different parts of the gold standard data are kept 'unknown'. In section 4.5, we will provide details on our evaluation methodology for the specific classification method that we use.

4.2 Data

Since no Dutch corpus annotated for animacy is available, the Dutch animacy annotation data from the Cornetto lexical-segmental database was used as gold standard noun data. A dictionary of nouns and their animacy status was extracted. The database consists of 40,392 word senses, but for the dictionary, all word senses with the same lemma were grouped and duplicates removed, since we will be doing type-based classification. All types where the different senses had different animacy categories, or no animacy value, were filtered out. Out of 31,959 types in total, 1008 had multiple different animacy categories (they were
Table 4.1: Selected examples of animacy categorizations in the Cornetto lexical-semantic database. All inanimate classes were taken together and ambiguous lemmas excluded.

<table>
<thead>
<tr>
<th>Human</th>
<th>Nonhuman</th>
<th>Inanimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Babander</td>
<td>ANWB</td>
<td>Groningen</td>
</tr>
<tr>
<td>Eerste-Kamerlid (Senate member)</td>
<td>appelboom (apple tree)</td>
<td>Koninginnendag (Queen’s Day)</td>
</tr>
<tr>
<td>afslammeling (descendant)</td>
<td>bremdeer (fire brigade)</td>
<td>appel (apple)</td>
</tr>
<tr>
<td>begeleidingsteam (coaching team)</td>
<td>cycoop (cyclops)</td>
<td>belastingkantoor (tax office)</td>
</tr>
<tr>
<td>drieling (triplets)</td>
<td>dienstensector (services industry)</td>
<td>compassie (compassion)</td>
</tr>
<tr>
<td>ex-burgemeester (ex-mayor)</td>
<td>embryo (embryo)</td>
<td>friettent (chips shop)</td>
</tr>
<tr>
<td>geallieerden (allied forces)</td>
<td>familie (family)</td>
<td>gebarentaal (sign language)</td>
</tr>
<tr>
<td>haantje-de-woorste</td>
<td>ijsbergsla (iceberg lettuce)</td>
<td>keel (throat)</td>
</tr>
<tr>
<td>juf (teacher (F))</td>
<td>meafteslaring (brined herring)</td>
<td>orkaan (hurricane)</td>
</tr>
<tr>
<td>oermens (primeval man)</td>
<td>microbe (microbe)</td>
<td>robot (robot)</td>
</tr>
<tr>
<td>racist (racist)</td>
<td>olfiant (elephant)</td>
<td>snelteijn (express train)</td>
</tr>
<tr>
<td>tachtiger (octogenarian)</td>
<td>snackbar (snack bar)</td>
<td>terrorisme (terrorism)</td>
</tr>
<tr>
<td></td>
<td>vrouwenrechten (women’s rights)</td>
<td>zeeuwer (seaweed)</td>
</tr>
</tbody>
</table>

| Table 4.1: Selected examples of animacy categorizations in the Cornetto lexical-semantic database. All inanimate classes were taken together and ambiguous lemmas excluded. |

ambiguous in terms of animacy. Furthermore, the annotation scheme, which is discussed in section 2.4.1, was simplified to match our classification categories of animate, nonhuman and inanimate by replacing specific inanimate subcategories by their parent animacy category, inanimate, in the dictionary. The institution category, which 1173 out of 40.392 word senses were labeled with, was changed to nonhuman. In the original hierarchy it is a category of its own, but since institutions have some animate-like linguistic properties, we thought nonhuman was a good fit. After these changes, the dictionary consisted of 5.311 nouns labeled as human, 1.908 nonhuman, and 23.732 inanimate.

The category definitions used in this database mostly seem to have a biological basis, which may not be ideal for many linguistic tasks. The nonhuman animate class is also very broad. Table 4.1 shows some selected examples of words in each class, from the data of Cornetto. None of these examples have ambiguous animacy (i.e. multiple word senses belonging to different animacy classes) in the database. We can see that the human class indeed contains mostly human entities, as well as some groups of humans. The nonhuman class is much broader, containing also groups of humans — family, fire brigade or services sector, the latter two could also be institutions or organizations. ANWB is also an organization. We also find some biological animates with various levels of sentience — apple tree, iceberg lettuce, microbe, elephant, and even a dish (brined herring) made out of a biologically animate entity. An abstract entity (women’s rights) also ended up in the category, as well as snackbar, whose near-synonym friettent resides in the inanimate category. This category also contains a possible institution (tax office), as well as many expected nouns. Some possible biological animates (apple, seaweed) also appear, as well as vehicles (express train) and intelligent machines (robot) which are said to
have some animate properties, and are indeed classified as ‘other animates’ by the Stanford-Edinburgh annotation. There is also a force of nature (hurricane) in this category, a class that is considered animate in some languages. Overall though, this class is more consistent in the data than the **nonhuman** class.

### 4.3 Context data

In a classification task, items to be classified are represented as a set of features (stored in a feature vector). These features are chosen to relate to the classification problem at hand - for example, for animacy classification of nouns, the features could be the verbs that the noun is in a dependency relation with in a corpus. However, no such context information is available in the dictionary of nouns that we have extracted from the Cornetto lexical-semantic database. To obtain it, we must look in an (annotated) corpus of texts, to see the nouns in use. The linguistic annotation then allows us to extract linguistic context information, such as subject relations.

In order to obtain such features for our classifier, the Lassy Large corpus was used (Van Noord et al., 2009). It’s a collection of syntactically annotated Dutch sentences from texts, such as newspaper articles. Full syntactic dependency trees, including dependency roles such as ‘subject’, are present in the annotation. This corpus consists of about 1.5 billion words, and the sentences have been parsed automatically by the Alpino parser for Dutch. This means that no human has checked the correctness of the sentence parses, and that they may contain some errors. However, this parser is the state of the art for Dutch.

This corpus lets us extract linguistic context features of nouns. The animacy-annotated nouns from the Cornetto data were looked up in this corpus, and certain types of dependency relations in which they appear were extracted, to be used as features in machine learning. For example, for each noun we counted how many times it occurred in a subject relation with specific verbs, information that the machine learner could use to determine animacy status. The features that were used will be discussed in chapter 5.

As an example, table 4.2 shows some subject relation information of the verb *schrijf* (to write), extracted from this corpus. It contains the verb and noun, their role (always *su* - subject, in this case), the construction in which the relation occurs, and the frequency of this relation. In this case, the frequencies are counted separately for each construction (i.e. transitive, intransitive), which can be useful in some cases, but it is also trivial to sum them together if necessary.

The full list of subject nouns for this verb ‘to write’ is, of course, full of names of people and organizations who have written something, like writers and politicians. It is a word that mostly takes human nouns in the subject role. However, we also find some exceptions, such as ‘article’ or ‘newspaper’, which can be explained by constructions like “The article says/writes that ....” In Dutch the verb for writing is apparently used in this sense. There are also
Table 4.2: Subject dependency relations of the verb *schrijf* (to write), extracted from the Lassy Large corpus.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Verb</th>
<th>Construction</th>
<th>Role</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>schrijf</td>
<td>intransitive</td>
<td>su</td>
<td>Amerikaan (American)</td>
</tr>
<tr>
<td>17</td>
<td>schrijf</td>
<td>intransitive</td>
<td>su</td>
<td>artikel (article)</td>
</tr>
<tr>
<td>15</td>
<td>schrijf</td>
<td>transitive</td>
<td>su</td>
<td>econoom (economist)</td>
</tr>
<tr>
<td>8</td>
<td>schrijf</td>
<td>np_np</td>
<td>su</td>
<td>fan (fan(person))</td>
</tr>
<tr>
<td>6</td>
<td>schrijf</td>
<td>transitive</td>
<td>su</td>
<td>hoofdperoon (main character)</td>
</tr>
<tr>
<td>48</td>
<td>schrijf</td>
<td>sbar</td>
<td>su</td>
<td>Le Monde</td>
</tr>
<tr>
<td>40</td>
<td>schrijf</td>
<td>intransitive</td>
<td>su</td>
<td>mens (human)</td>
</tr>
<tr>
<td>11</td>
<td>schrijf</td>
<td>np_ld_pp</td>
<td>su</td>
<td>Mozart</td>
</tr>
<tr>
<td>5</td>
<td>schrijf</td>
<td>sbar</td>
<td>su</td>
<td>Oscar Wilde</td>
</tr>
<tr>
<td>1</td>
<td>schrijf</td>
<td>transitive</td>
<td>su</td>
<td>zon (sun)</td>
</tr>
</tbody>
</table>

quite a few seemingly senseless entries, such as 'sun', but these often have a frequency of 1 which indicates that they might be the result of some sort of parsing error, or even a spelling error in the original text — maybe *zon* (son) was intended for *zon* (sun). Since we are using a statistical machine learning method, erroneous low-frequency outliers should not affect the final result too much.

Another type of context feature are morphosyntactic distributional features, as used by Øvrelid (2009). These features are counted over the entire corpus for each noun, for example, how frequently a noun occurs in a subject relation or in an object relation, without taking into account which specific verb it is an object or subject of.

The kinds of features described in this section should be able to provide a classifier with information regarding the animacy of a noun. In the next section, I will discuss the classification algorithm that can turn this extracted information about nouns into an animacy class label.

### 4.4 Memory-based learning

Like in the Swedish animacy classification project of Øvrelid (2009), we make use of the k-nearest neighbour (KNN) algorithm as implemented in TiMBL (Daelemans et al., 2007). This algorithm, also known as memory-based learning, is a supervised machine learning method that compares feature vectors of novel items to those of items for which the class is already known. It then bases its classification decision on the class of the *k* nearest items (in terms of feature similarity), where *k* is any number of neighbouring items. Like most modern classification algorithms, this is a probabilistic approach that bases its classification on data, rather than on expert knowledge.

Natural language processing tasks were traditionally often performed with a
knowledge-based approach, using predefined knowledge such as rules and grammars to perform tasks like parsing or classification. Since the 90s, empirical methods using statistics derived from corpora of text data have become more popular. These probabilistic methods have the advantage of greater coverage of linguistic data, being more robust, being less specific and taking less time to develop. Knowledge needs to be specifically tailored to the domain it is to be used for, while statistical methods are less domain-specific and can be re-used. The K-nearest neighbour algorithm can be used for all kinds of machine learning problems and does not involve any specific knowledge about animacy, apart from the selection of features for it to use.

Probabilistic NLP systems gain their knowledge from example training data. While a rule-based system needs human experts to define rules or heuristics that accurately capture the task, a probabilistic system can infer this information from the data in some way. In an unsupervised system, all of the data is unlabeled and the correct classes are not known, so there is no way to evaluate the result, or apply meaningful class labels (such as 'inanimate'). In a supervised probabilistic system, there is data available that has been classified correctly (i.e. for animacy classification, the correct animacy class is known). By detecting patterns in the features (properties) that are associated with each noun class, the system can associate features or combinations thereof with specific classes. The K-nearest neighbour algorithm is a bit different here in that the typical notion of 'training' doesn't really apply. Most probabilistic classifiers, such as those based on Hidden Markov Models, do some kind of abstraction. HMMs aggregate instances into probabilities, without preserving details about individual items, such as outliers. This is known as eager learning, where infrequent and exceptional cases are filtered out. This is not always desirable, since such tasks often have many exceptions that should be included, and abstraction can be harmful to performance. In contrast, KNN preserves all training instances in its 'model', without doing any generalization. This is known as lazy learning, where all data is stored and no abstraction is performed on it (Daelemans and Van den Bosch, 2005).

Many probabilistic methods can have problems with sparse data, when there is not enough to accurately calculate a probability. They may have problems determining the relevancy of data instances, and it's not really clear what these methods do internally so it can be hard to interpret them, explain what they do, or check them for errors. Since KNN does not do any generalization, it does not have this problem, as there are no 'hidden' states or abstracted elements whose origin cannot easily be deduced. Sparse data may still be a problem, but at least instances with little associated information will not be generalized away, since all instances are kept. KNN is also able to handle symbolic features rather than numeric ones, which is relevant for some NLP tasks, though not so much for our method of animacy classification, where we use (numerical) association values as feature values.

KNN, or memory-based learning, is claimed to fit the properties of natural language processing tasks very well. It is a nonparametric method, which means that it does not make any assumptions about the distribution of the data. For example, in Gaussian Mixture Models, the data is assumed to be normally distributed in some way, so that it can be fitted to one or more Gaussians. KNN
Table 4.3: Example feature vector for KNN animacy classification. Contains three training nouns: bestuurder (driver, director), parool (parole (word of honour), a newspaper), VVV (tourist information offices).

<table>
<thead>
<tr>
<th>Noun</th>
<th>komt_binnen</th>
<th>laaid</th>
<th>Animacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>bestuurder</td>
<td>0.000027447</td>
<td>0.9899999999</td>
<td>human</td>
</tr>
<tr>
<td>parool</td>
<td>0.7301998638</td>
<td>0.0002008943</td>
<td>nonanimate</td>
</tr>
<tr>
<td>VVV</td>
<td>0.4287437089c</td>
<td>0.4504258805</td>
<td>nonhuman</td>
</tr>
</tbody>
</table>

makes no such assumptions, which is good for many NLP tasks since many natural language phenomena have highly skewed distributions. For example, word frequencies in language follow a Zipfian distribution, with a high number of low-frequency words, and a long ‘tail’ of high-frequency words, which are much more rare.

To have ‘neighbours’ in nearest neighbour classification, it is necessary that items of data can be positioned in some feature space, in which one can speak of distances to other items (and therefore neighbours). Data must be in metric space and their distances can then be compared with a distance metric, which will be discussed later in this chapter. After all, one cannot just say that a word is close to another one without having some objective measure of comparison. In our case, words are compared by the similarity of their associations with verbs in object, subject or adjective relations, which are stored in feature vectors as numeric values. The algorithm can also run on categorical data, in that case, a number could be assigned to each category, for example.

A memory-based classifier consists of two parts, analogous to the ‘training’ and ‘testing’ phases that all statistical NLP algorithms have. As discussed earlier, the training part is memory-based and involves very little processing of the training instances. The testing, or classification phase, then consists of comparing novel data items to the items in memory to obtain a classification decision. The item is compared to its ‘nearest’ neighbours, in terms of the feature space. The next section will illustrate this with an example.

### 4.4.1 Training and testing

As described earlier, the training phase involves simply storing the training data in memory. We will now present a simplified example, in which animacy classification takes place on the basis of a noun’s subject relations with two specific verbs, binnenkomen (to come in) and laaiden (to sound/say/read). The first verb is expected to mostly take animate subjects, while the second is likely to take inanimate ones, so we expect to be able to make a decent distinction on the basis of the relationships of nouns with these specific verbs. Therefore, these two verbs are the features. As feature values, we will take the statistical association strength between each noun and the feature verbs, i.e. whether they occur together more often than would be expected by chance. This method will
a

- Noun	 kom_binnen	 luid	 Animacy
- \textit{dracht} 0.8461477240 0.1180349545 ???

Table 4.4: Example feature vector of a novel word for KNN animacy classification. Contains the noun: \textit{dracht} (attire, gestation), which did not occur in the training data and the animacy class is therefore unknown to the classifier.

be explained in section 5.2.1. This leaves us with the training feature vector in table 4.3.

The classifier then stores this data in memory, and this is the model. No generalization takes place. New data items, for which the animacy class is not known (i.e. table 4.4), can then be compared to this model to classify them. Figure 4.1 illustrates the concept. It shows a two-dimensional feature space of the two verb-subject relations. Normally there would be more verb-subject relation features and therefore more dimensions, but this is an example. Training nouns are positioned in this space depending on their association strength with the verbs (our feature values). For these training nouns, the class is known — the grey dots represent inanimate nouns, the red ones are human animates, and the green ones are nonhuman animates.

A new item can be compared to one or more (1-k) neighbours in the feature space, hence the name of the algorithm. In the example, for the novel word \textit{dracht}, the nearest neighbour in this simplified two-dimensional feature space is the word \textit{parool}, which is known to be inanimate in the training data. And when the nearest neighbour is inanimate, the new item is also classified as inanimate.

In figure 4.2, we have added some more nouns and increased \( k \) to 5. Now, the 5 nearest neighbours of \textit{dracht} are taken into account for classification. We see that the single nearest neighbour is a human animate word, but this must be an outlier or unusual word, since the remaining 4 of the 5 nearest neighbours are inanimate, and thus our novel word is classified as inanimate.

Apart from the number of neighbours, there are a few other details to be explained here. A similarity measure is needed to measure the distance between a noun and its neighbours in the feature space, i.e. how similar they are. Furthermore, some form of feature weighting is often used in KNN. While it is not obvious from this simple two-dimensional example (2 features), when there are many features it may be the case that some features are more informative than others for making a classification decision. For example, in animacy classification it could be that there are some verbs that have no selection restrictions for the animacy of their subject at all. Feature weighting can automatically decrease the importance (weight) of such irrelevant features, without having to do any sort of manual selection of verbs. A third detail of this method is class voting — how are the classes of the \( k \) nearest neighbours used to make the classification decision? One could simply take a majority vote, where the class with the most near neighbours wins, but there are also other options, such as weighing the closer of the 5 neighbours more heavily for the classification.
decision. In the next sections, I will explain these details further.

4.4.2 Similarity measures

In k-nearest neighbour classification, the way in which a novel item is compared with its neighbours can vary. The Timbl memory-based learner includes various distance metrics, or similarity measures, that can be used for this purpose. A similarity measure can determine the distance between a novel instance $X$ and a (memorized) training instance $Y$ ($\delta(X, Y)$), measuring distance in the feature space, where $X$ and $Y$ are represented by their features (verb argument relations, in our case). A basic similarity measure included in Timbl is the overlap metric (Daelemans and Van den Bosch, 2005, p. 29), which simply measures the sum of the differences between the features. Using numeric features, such as the association scores from our example, the metric is defined as such:

$$\sum_{i=1}^{n} \frac{|x_i - y_i|}{\max_i - \min_i}$$

where $x_i$ is a feature of $X$ and $y_i$ is a feature of $Y$.

Taking our previous example, we get the following distance between *dracht* and the inanimate *parool*:

$X = \text{dracht}$

$Y = \text{parool}$
Figure 4.2: Visualization of a two-dimensional feature space for KNN animacy classification with $k=5$

$$i = (\text{kom\_binnen, luid})$$

$$\frac{[0.8461477240 - 0.7301989638]}{1-0} + \frac{[0.1180349545 - 0.0000027447]}{1-0} = 0.23378282$$

While the distance between \textit{drecht} and the animate \textit{bestuurder} is much greater:

$$X = \text{drecht}$$

$$Y = \text{bestuurder}$$

$$i = (\text{kom\_binnen, luid})$$

$$\frac{[0.8461477240 - 0.0000027447]}{1-0} + \frac{[0.1180349545 - 0.9999999999]}{1-0} = 1.72811002$$

Similarity measures such as this one allow us to measure which neighbours are the nearest in a feature space. Timbl includes several other such measures:

**Modified value difference (MVDM)** This metric is designed mainly with symbolic features in mind, in which some values are more similar than others. It determines how similar these values are by looking at their co-occurrence with target classes. Values with more different conditional distributions (for the classes) are considered to be further apart in the feature space (Daelemans and Van den Bosch, 2005, p. 38). This method therefore seems somewhat irrelevant for high-precision numerical features such as the ones in our example.

**Jeffrey divergence** This measure is similar to MVDM in that it also computes the distance between class distributions of feature values. However, it
includes a logarithmic term rather than MVDM’s geometrical distance. Because of this, it assigns relatively larger distances to distributions that differ more, which emphasizes zero-probabilities, making this metric better at handling sparse data (Daelemans et al., 2007).

**Levenshtein metric**  This is an edit distance metric specifically for strings. It counts the number of deletions, insertions and substitutions required to transform one string into another, and can therefore serve as a measure of how different two strings are. This metric is useful when feature values are words, not numeric, like our example (Daelemans et al., 2007).

**Dice coefficient**  The Dice coefficient as implemented in Timbl compares feature value strings, like the Levenshtein metric. Rather than computing edit distance, it computes the overlap in character bigrams for two strings (Daelemans et al., 2007).

**Dot-product metric**  The dot product is a general operation for vectors, that can also be applied to numeric or binary feature vectors. The dot product is higher for better matches, and is subtracted from the maximum possible dot product (that of an exact match) to obtain a distance metric. This metric is better than Overlap for sparse data, since matching zero values are not counted (Daelemans et al., 2007).

**Cosine metric**  Another vector operation, similar to the dot product. It basically normalizes the dot product of two vectors by their length, to compensate for large differences (Daelemans et al., 2007).

While the metrics differ enough from each other to be able to choose one based on their properties, we instead rely on Wrapped Progressive Sampling (WPS) to choose the best similarity measure. WPS is an algorithm that can compute optimal parameters through repeated testing, selecting the parameter settings that result in the highest accuracy scores. This method will be explained in section 4.4.5.

### 4.4.3 Feature relevancy

Usually, some features of a classifier are more informative than others, particularly when no feature selection has taken place. We are using relations of nouns with high-frequency verbs as features, but some verbs may not select for animacy at all, and may not be able to contribute much to the task of animacy classification. To quantify this, feature weights can be automatically determined with some kind of relevancy measure. Features that are more relevant can then be weighted to contribute more to the classification decision. The goal of these feature relevancy measures is to determine which features are good predictors of the class labels. They are computed over the training data in memory, and could therefore be seen as a sort of training phase, which KNN otherwise lacks. Timbl contains various feature relevancy measures:
**Information gain** One such measure, based on information theory, is that of information gain. This measure is useful for deciding on splitting criteria in decision trees (i.e., which split will lead to the biggest information gain), but may also be used for feature relevancy. For each feature, it estimates how much it contributes to the known class label from the training data, by measuring the difference in entropy that having knowledge of this feature causes. Entropy is a measure of uncertainty, i.e. about what the correct class label should be. It is computed as follows:

\[
    w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \tag{4.2}
\]

where \(C\) is the set of class labels, \(V_i\) is the set of values for feature \(i\), and \(H(C)\) is the entropy of the class labels:

\[
    H(C) = -\sum_{c \in C} P(c) \log_2 P(c) \tag{4.3}
\]

Because this measure assumes a set of values, numeric values are discretized into a limited number of different values before computation (20 by default). (Daelemans and Van den Bosch, 2005, p. 29)

**Gain ratio** This is a variation on Information Gain that normalizes over the entropy of the feature values. This is to solve a weakness of Information Gain, where it tends to overestimate the relevance of features with a large number of values (Daelemans and Van den Bosch, 2005, p. 30). Since we are using numeric values that are discretized anyway, there should be little difference, but we leave the choice up to the WPS algorithm.

**Chi-squared** The \(\chi^2\) statistic can also be used for feature weighting. Its main advantage is that it can be compared across features with different degrees of freedom, preventing the bias towards features with more values that Information Gain and Gain Ratio have. For numeric features, discretization is also applied here (Daelemans et al., 2007).

Despite the possibility of feature weighting reducing the weight of irrelevant features, it may still be better to leave features that are known to be irrelevant out entirely. We have not explored this option however — since this work is the first to use this kind of feature set for animacy classification, we do not know which features might be irrelevant. Making such a selection manually would also introduce domain knowledge or language-specific knowledge into a process that is otherwise language-independent (though resource-dependent). Again, we do not make a manual choice for a metric here, we instead use the WPS algorithm to choose the optimal metric for each experiment.

**4.4.4 Class voting**

The final aspect of KNN classification that we will discuss here is class voting. In cases such as our example 4.2, where \(k\) (the number of neighbours) is greater...
than one, multiple neighbours contribute to a classification decision, and they can do so in different ways. The simplest way is majority voting, in which we simply take the majority class among the $k$ nearest training examples. However, there are other options that might be of interest, particularly for higher values of $k$. We have already seen that a low value of $k$ may lead to interference of noisy data, for example if there is one outlier that happens to be the nearest neighbour. However, for larger values of $k$ a relatively large area of the feature space may be taken into consideration, especially when data is sparse. In standard majority class voting, the more distant neighbours in this area and the nearest one contribute equally, even though there may be a large difference between them. Alternative class voting methods may weigh the votes by the distance to the item to be classified. The following are available in Timbl:

**Inverse-linear distance**  This scheme weights nearer neighbours more heavily, by dividing the relative distance of a neighbour by the relative distance of the nearest neighbour as such:

$$w_j = \frac{d_k - d_j}{d_k - d_1}$$  \hspace{1cm} (4.4)

where $d_j$ is the distance to the $j$th nearest neighbour, $d_1$ the distance to the nearest neighbour, and $d_k$ the distance to the furthest neighbour out of the $k$ neighbours that are taken into consideration for voting. So, the nearest neighbour gets a weight of 1, the furthest neighbour gets a weight of 0, and everything else is linearly scaled between them (Daelemans and Van den Bosch, 2005, p. 43).

**Inverse distance weight**  This metric is simply the inverse of the distance to the neighbour, plus a small constant in the denominator to avoid division by zero. The authors state that, in empirical tests, this metric was usually outperformed by the inverse-linear distance (Daelemans and Van den Bosch, 2005, p. 43).

**Exponential decay**  In this metric, the weight decreases exponentially with distance, and two constants determining its slope and power can be set to adjust the rate of decrease (Daelemans and Van den Bosch, 2005, p. 43).

Again, we leave it up to the WPS algorithm to determine whether to use majority voting or one of the weighting metrics. The next section will discuss this algorithm.

**4.4.5 Wrapped Progressive Sampling**

Wrapped progressive sampling is a method for automatically finding optimal classification parameters on a given training set. In Timbl, there are over 900 possible parameter combinations, which makes it difficult to manually choose a good combination of settings. In addition, Van den Bosch (2004) have shown that an average of 31.2% error reduction could be achieved on various classification problems by automating this process, compared to using the default
settings. While this is not conclusive evidence (maybe the default settings are not good), it shows that there may be something to be gained from empirical estimations.

The method should be used for optimizing parameters on the training data, with known class labels. Classifier wrapping involves the use of an internal training and test set to avoid overfitting and estimate performance on new data. Instances of the classifier with different parameter settings are thus trained on one set and tested on the other, just like in a regular experimental setup. The accuracy on the test set is then taken to be the performance of that particular combination of settings.

Progressive sampling refers to the fact that the settings are trained and tested on increasingly larger samples of data, as the number of possible optimal settings decreases. Initially, a sample of 500 instances is used for training, with 100 testing instances, and the entire space of possible settings is tested on them. Then, a selection of the best performing settings is made, and they are evaluated again on a larger set of instances. This process continues until the entire set of instances (i.e. the training data) is used, or until only one setting remains.

To select which settings to keep, the settings are divided into 10 bins that each represent an equal proportion of the range of accuracies found for the settings. The first bin (with the best accuracies) is always selected, as well as any remaining bins that have equal or more settings in them than the subsequent bin. This method should capture a group of well-performing settings, even if it is larger than one bin.

With this method, a large parameter space can be searched without having to run a large number of tests (with a training and testing phase) on large datasets. This speeds up the parameter setting process.

4.5 Evaluation method

In the next chapter, we will discuss each feature that we used for our classifier, and evaluate their contribution to solving the animacy classification problem, and in the subsequent chapter, we will provide a general evaluation of the system. Therefore, we will now explain how our accuracy scores in these evaluations are obtained.

We have already discussed the particulars of training and testing in k-nearest neighbour classification. In training, data examples are memorized, and in the testing phase, novel items are compared to the training examples for classification. This means that for a test, we need distinct training and testing data sets, otherwise, classification would be trivial. For evaluation, we split up the portion of the nouns that we will use in the classification task. We do not always use all of the nouns. In most experiments, we use a frequency cutoff of 10 to exclude nouns that occur rarely or never in the Lassy Large corpus. So to be included in the dataset, a noun must occur in a relevant dependency relation (that we use as a feature) 10 times or more. Low-frequency nouns are very difficult to classify, since there is not much feature information available for them. Classifier
Table 4.5: Example of a confusion matrix. Columns are classes predicted by the classifier, and rows are actual noun classes. **Bold** values are correct predictions.

<table>
<thead>
<tr>
<th></th>
<th>Human</th>
<th>Nonhuman</th>
<th>Inanimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>151</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>Nonhuman</td>
<td>0</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td>Inanimate</td>
<td>1</td>
<td>3</td>
<td>982</td>
</tr>
</tbody>
</table>

performance for nouns of different frequencies will also be evaluated in the next chapter. There is also a large portion of nouns in the Cornetto data (14,496 out of 30,950) that are unknown in the Lassy data, possibly due to differences in lemmatization or the way compound nouns are handled in the two different data sources.

After this frequency cutoff, the remaining nouns are then split into 10 sets of equal size and with the same balance of the three animacy classes as the full data set. This allows us to perform a ten-fold cross validation, in which the classifier is trained and tested not just once, but ten times, in order to obtain an accuracy score that is an average of ten tests, which is more reliable than using just a single test. With this method, each of the 10 sets is once taken as the test set, while the remaining sets are used for training. This ensures that each noun is used for testing, and only once, while still making sure that no test data is ever included in the training set. We report the average accuracy from these 10 tests as the accuracy score in our evaluations.

In some cases, we will also look at the classifier’s **confusion matrix** (or error matrix). This is a table that counts the actual class of the nouns on one axis, and the class predicted by the classifier on the other. This lets us examine the classifier’s performance for each class, and shows which classes are commonly ‘confused’. An example is shown in figure 4.5. This matrix is for a classifier that never predicts the **nonhuman** class (and therefore gets all words of this class wrong). Instead, all nonhuman nouns are classified as inanimate. In contrast, almost all of the inanimates are classified correctly, and human nouns are classified with 84% accuracy (151 out of 175).

Before each evaluation, we first determine the optimal Timbl-parameter using the wrapped progressive sampling method described in the previous section. We then run the evaluation with those parameters, even if we might be able to set better parameters manually.

### 4.5.1 Baseline

Since animacy classification for Dutch has not been done before to our knowledge, and it is difficult to compare the scores to other animacy classifiers that work differently, we start from a very simple baseline. Our baseline is the accuracy score that can be obtained by classifying every noun as the majority class
(normally inanimate). In the full data set, 76.68% of the nouns are inanimate, but this proportion changes when different frequency cutoffs are used.
In this chapter, we will discuss the features upon which our animacy classification is based. In chapter 3, we have already discussed features used in existing animacy classifiers, we will summarize this here, and then proceed to elaborate on the features we have chosen to use.

To know whether a word refers to an animate or inanimate entity, we need to have some information to infer this from, even as humans. We might simply know it from previous experience somehow — if I’ve talked to an interpreter before, I know that it’s a human who does language interpretation. It is part of our knowledge. We could also infer it from the morphology of a word — if a fireman and a businessman are human, then a longshoreman probably also is, even if we haven’t heard of the profession before. Lastly, we could infer it from context. If we read “The wug pondered the issue and decided to run”, we have a good clue that a wug is at least fairly sentient, and also able to run. On the other hand, if we read “The wind has blown the bicket down” we can at least infer that a bicket is a pretty passive entity.

For natural language processing purposes, we can use databases of knowledge assembled by humans as a replacement for ‘background knowledge’. Earlier we discussed an animacy classifier where this kind of features was used. Orasan and Evans (2001) used the hierarchy of WordNet to obtain information about animacy as a semantic class for a large number of the words available in the database. Based on the animacy property of the ‘unique beginners’ (such as animal or person) at the top of the hierarchy, they infer the animacy of word senses lower in the hierarchy. They also have a similar bottom-up method, starting from a sense-annotated corpus. In these classifiers, the ‘background knowledge’ from the annotation directly serves as the classification features. The main drawback of this approach is that it cannot handle unknown cases (i.e. cases that are not in the knowledge base).

The second approach, examining the morphology of the noun, was discussed in Baker and Brew (2010). They are classifying Japanese, which can have suffixes indicating animacy. They incorporated this information not as a feature in their classifier, but as a pre-processing step to make the task easier. They grouped nouns ending in the same character (suffix) into equivalence classes,
treating them as single items in classification. An English example would be
to group all nouns ending in -man together for feature extraction and for the
classification. However, this approach is language-specific and only works when
a large amount of nouns have a clear compound structure, with not too many
exceptional cases (i.e. the cayman or daman, which are animals). Their actual
features are similar to what we will be using, though their feature values are
different.

The third approach is to use context information. We have already discussed
why classifiers generally use distributional contextual features, i.e. taking the
contexts of every occurrence of a noun into account, rather than just one, which
does not provide enough information. The use of this sort of context informa-
tion proved to be successful in previous work. Øvrelid (2009) made use of
grammatical features, for example, counting the use of the genitive for a noun
in a corpus, or the ratio of subject vs object roles of a noun. Such features are
easily obtained from any corpus in which those properties are annotated, such
as dependency treebanks. We have also tested Øvrelid’s most successful feature
in our classifier, the object-subject ratio, which will be discussed in the next
section. Their other features did not perform as well in the evaluation, possibly
because they are not general enough. For example, not every noun occurs in
the genitive case or in a passive construction, while almost every noun will at
some point show up in a corpus as either an object or a subject of a verb.

A more detailed approach to context information was taken by Baker and
Brew (2010), in a distributional semantics sort of approach. Rather than coun-
ting noun subject/object roles in general, they counted noun subject/object roles
with specific verbs, i.e. how often driver is the subject of to drive. Then, each
verb is a feature and those frequency counts are the feature values. Alternative
feature values were also evaluated — the Verb Animacy Ratio, the number of
animate subjects/objects divided by the total number of subjects/objects for
that verb in the training data, rather than just the total frequency. However, we
see some difficulties with generalization to novel data with this approach. The
animacy ratio will always be based on the training data, even if the animacy
ratio of verbs for a different corpus or document is completely different. That
information cannot be known without actually having animacy annotation. We
also question the usefulness of relying on knowledge of verb animacy (derived
from noun animacy) for the task of determining a noun’s animacy, and we would
prefer to avoid such additional data requirements. In our classifier, we use this
type of feature based on distributional semantics as well, but for feature val-
ues, we only use frequency counts and statistics derived from them, as will be
explained in this chapter.

5.1 Object/subject frequency of verb relation

Firstly, we evaluated the most successful feature of Øvrelid (2009) for our sys-
tem, which was the object/subject ratio, i.e. the relative frequency of object
dependency relations with verbs, as opposed to subject dependency relations.
Whenever a noun occurred as the object of any verb in the Lassy corpus data, it
was counted as an object relation, otherwise as a subject relation. The feature
value is calculated as follows:

\[ osratio(N) = \frac{f(N_s)}{f(N)} \]  

(5.1)

where \( N \) is a noun, \( N_s \) is a noun in a subject relation, and \( f \) denotes frequency.

Using the relative frequencies as two separate features (one for objects and one for subjects) was also tested, but gave the same results. There was no balancing for the total amount of object relations and subject relations (there are more subject relations in the data). We did perform another experiment where only objects and subjects of transitive verbs (i.e., verbs that take both an object or a subject) were counted. In this situation, object and subject counts should be balanced, though it excludes a portion of the data. Only nouns with a frequency of 10 or more in the Cornetto data were used in these experiments.

Table 5.1 shows the results using this feature with WPS-optimized parameters. The baseline is a classifier that classifies every noun as the majority class, inanimate. A look at the confusion matrix of this evaluation shows that this is also exactly what our objsubj-based classifier does. It is unable to distinguish the categories and therefore never even tries to guess human or nonhuman. The small difference with the baseline is due to the way the data is split up into 10 sets. Objsubj-trans is the transitive experiment, which performs better, but still barely reaches above the baseline. This contrasts with the results of (Øvrelid, 2006), who reached accuracy scores well over 90%, though using a smaller set of nouns that occurred with greater frequency and with a higher baseline, which makes the task easier. In addition, they performed a two-way classification of the animate and inanimate classes, while we are deciding between three classes — human animate, nonhuman animate and inanimate. In our experimental setup, this object-subject ratio feature does not seem to be sufficient to make the distinction.

### 5.2 Distributional lexical features

Instead of just using object or subject positions as a feature, we decided to be more specific, and look at object and subject dependency relations with specific verbs as well. For example, rather than having “bee is an object 80% of the
time" as a feature, we will instead say "tree is an object of to hit 80% of the time".
More generally speaking, we collect features based on the lexical items that a noun has dependency relations with. The following example will illustrate this:

This sentence contains the inanimate noun banana, however, the sentence somehow doesn't make much sense. Banana is in a subject relation with to think, i.e. the sentence states that the banana is thinking. Furthermore, banana is modified by the adjective thrifty, i.e. the banana is a thrifty banana. If this sentence said trader instead of banana it would be fine, but as it is, it violates the semantic selection restrictions of the adjective thrifty and the verb to think. These selection restrictions are generally not hard rules, but they are certainly tendencies that can be identified from frequencies of such relations in a corpus. The verb to think is much more likely to take highly animate and sentient subjects (i.e. humans), though there seems to be no such tendency for its objects (one can think about anything). Similarly, thrifty is a property that is generally associated with animate beings and at least requires the ability to think about resources. It is generally not applicable to bananas. Thus, it should be possible to make inferences about a word's animacy based on the dependency relations it occurs in, in a large corpus. With the Lassy Large corpus available to extract such features from, it should be possible to collect enough of such distributional data for relations with the most common lexical items at least.

This approach builds upon ideas from distributional semantics. According to this view, words that are similarly distributed (i.e. occurring in similar contexts), also have similar meanings. This idea was explored by Hindle (1990), who used a similar methodology of examining subject and object relations with verbs for semantic noun classification (in terms of semantic similarity). He defines noun similarity in terms of the mutual information (a measure of association) of verbs and their (noun) arguments. Our approach follows this basic idea, and we also use association measures to determine which verbs (or adjectives) are typical for a noun.

As mentioned before, Baker and Brew (2010) also use this type of feature, but with different feature values. Because they perform different (and language-specific) preprocessing, and because they are mainly working on Japanese which is quite different (i.e. better suited to morphological preprocessing), it is difficult to compare both approaches directly.

5.2.1 Feature value

The distributional semantics work by Hindle (1990) shows that mutual information can be used to quantify the association between nouns and their dependencies. However, since we are only interested in capturing information about animacy classes, rather than the much more fine-grained notion of semantic
classes or semantic similarity, maybe a simpler measure is sufficient. We tested this by evaluating the classifier with various measures for the feature values.

**Binary**

The most basic metric is a binary one, where we simply use the value 1 if the noun occurs in this dependency (i.e. as a subject of a specific verb), and list 0 otherwise. However, it does not encode any sort of weight or importance. If a dependency occurs just once, it will have the same importance as a highly frequent dependency. A single occurrence could happen in an unusual context or even as a parsing error, which is quite possible since the corpus from which we extract features was automatically parsed. Classification with just binary features is much faster, though. Rather than having an infinite range of possible values, features can have only two possible values, which makes the classification process computationally easier, for example the calculation of the overlap measure. The results (table 5.2), however, show that this metric seems to be sufficiently informative, and the classifier with binary feature values is only slightly outperformed by more advanced measures. Surprisingly, it looks like binary features are a good alternative for speedy classification. However, it should be noted that even scores of around 90% may imply a large number of errors for some cases, this will be explored in section 6.2.

**Frequency**

More information can be captured by taking the frequency of the dependencies. This is less efficient, but now, more common dependencies have higher feature values and this will affect the neighbour distance calculation. Thus, if animate words occur a lot in a dependency with a specific lexical item, while inanimate words also occur with it, but only rarely, the classifier will be able to make that distinction. This value seems to work surprisingly well (table 5.2). We believe that it introduces a bias towards dependencies involving common words, but the classifier performs well with these settings nevertheless.

<table>
<thead>
<tr>
<th>Feature metric</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>80.92%</td>
</tr>
<tr>
<td>Binary</td>
<td>92.19%</td>
</tr>
<tr>
<td>Frequency</td>
<td>93.09%</td>
</tr>
<tr>
<td>Normalized Frequency</td>
<td>89.23%</td>
</tr>
<tr>
<td>Pointwise Mutual Information</td>
<td>92.27%</td>
</tr>
<tr>
<td>Fisher’s Exact Test</td>
<td>91.37%</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of feature value metrics, using the optimal setup otherwise (explained in section 6.2).
Normalized frequency

The raw frequency measure may not be precise enough. For example, an uncommon dependency that occurs relatively frequently may be drowned out by dependencies involving words like man, just because they are highly frequent nouns, and any dependency involving them will probably occur a lot. We are not interested in the effect of these word frequencies, so we can reduce their influence by normalizing them out. We can divide the frequency of the dependency by the total frequency of the noun in the corpus to obtain a normalized frequency:

\[ \text{normfreq}(N_i, D_j) = \frac{f(N_i, D_j)}{f(N_i)} \]  

(5.2)

where \( N \) is a noun, \( D \) is a dependency feature, and \( F \) is frequency. However, the results show that this measure does not perform better. A possible cause may be the fact that this measure generates fractional numbers, and therefore an infinitely large amount of possible feature values. It may be better to have a limited number of possible feature values due to Timbl’s decision tree implementation and because some measures (i.e. information gain) involve discretization anyway. Having so many feature values is also less efficient, when calculations have to be performed for each of them.

Pointwise Mutual Information

A slightly more advanced method of normalization is to take the Pointwise Mutual Information (PMI) score of each relation. This is the same method that was used by Hindle (1990) in their work on distributional semantics. PMI is a measure from information theory that measures statistical association. It can be used to determine how much the probability of the dependency relation deviates from that of the probabilities of the noun and the related lexical item individually. It produces higher values for relations that occur more often than would be expected by chance. More informative relations will get a higher value, reducing the effect of the frequency of the noun and verb compared to the other method of normalization. PMI is calculated as follows:

\[ PMI(N_i, D_j) = \log \frac{p(N_i, D_j)}{p(N_i)p(D_j)} = \log \frac{p(D_j|N_i)}{p(D_j)} \]  

(5.3)

where \( p(N_i) \) is the probability of noun \( i \), i.e. its frequency in relations divided by the total number of relations. The results in table 5.2 show that this measure performs well, though not quite as well as the frequency measure. This turned out to be due to bad performance of the WPS parameter tuning — when manually setting the parameters, we were able to reach an accuracy of 93.34%. We report the WPS score to be consistent in methodology, but consider this method to be competitive as well.

Fisher’s Exact Test

This is another statistical association measure. Because PMI has been said to have problems with sparse data, which, like in most NLP tasks, we have a lot of,
we have also tried another measure. Fisher’s Exact Test has the advantage of being exact and does not make assumptions about the distribution of the data, which tends to be skewed in NLP tasks. For these reasons it has been used in similar tasks involving association of lexical items, such as collocation and collostructional analysis (Stefanowitsch and Gries, 2003). Because we would like to be able to express both dependence and independence, we use the two-tailed variant. A downside of the exactness of this method is that it is computationally intensive, but with the increase of computing power that is becoming less of a problem. The test expresses a p-value indicating the significance of an association, which is calculated as follows:

\[ P(N_i, D_j) = \frac{(a+b)(c+d)}{(a+c)(n-a)} \]  

where \( n = f(N, D) \) (the total number of relations) 
\( a = f(N_i, D_j) \) 
\( b = f(D_j) - f(N_i, D_j) \) 
\( c = f(N_i) - f(N_i, D_j) \) 
\( d = n - a - b - c \) 
and \((a+b)/a\) is calculated as \( \frac{a+b}{\log a} \).

This can be visualized as a contingency table, such as table 5.3. From the four bolded values — frequency of the relation, frequency of the noun, frequency of the verb and total number of relations — the other values can be calculated, such as \( b \) — the number of times that the dependent lexical item \( D_j \) occurs in a relation, but not with noun \( N_i \). These values are then used to obtain a p-score that represents how likely it is that these two lexical items are in a dependency relation due to mere chance. This represents the strength of the association, a low p-value indicates that they are likely associated. In the example of table 5.3 the p-value is less than 0.00001, indicating a strong association. In Dutch, the verb \( \text{ontstaan} \) is indeed commonly used to describe feelings coming up, which in turn is one of the few situations in which \( \text{gevoel} \) would be placed in a subject position. If the verb \( \text{ontstaan} \) also has such associations with other inanimate nouns in the subject position, it could be a useful animacy classification feature.

However, the Fisher’s Exact Test measure does not produce better results
than the Pointwise Mutual Information measure for our task. A possible explanation could be that this test does not measure effect size - it can say that the lexical items are dependent or independent, but isn't really designed to specify by how much. The p-value only indicates the probability of dependence or independence. This may cause loss of information compared to other association measures that do indicate the effect size, such as PMI.

For the remaining evaluations, we have used PMI as our feature value metric, unless otherwise specified. We consider it the best choice — the accuracy difference with other measures is minor and can be explained by imperfections in the automatic parameter tuning, and this measure should be more informative in theory — it mostly filters out the effects of word frequencies. We will now evaluate the dependency relation types that we used for animacy classification — verb subject, verb object and adjective relations.

\subsection{5.3 Verb subject relations}

In this section, we discuss dependency relations in which the noun is the subject of a verb. Nouns in subject positions commonly have the role of agent, i.e. it causes some sort of change, a role that animate entities often take. This was the idea behind the object/subject ratio feature. But this type of feature, subject relations with specific verbs, may also gain information from the selection restrictions of specific verbs. Compare:

\begin{enumerate}
\item[(24)] The \textbf{man} thought it looked strange.
\item[(25)] ? The \textbf{box} thought it looked strange.
\end{enumerate}

It is more difficult to find such an example of the other case, i.e. a verb that mostly takes inanimate nouns, but if there are verbs with such tendencies too, the classifier should be able to take it into account too.

We have evaluated this feature type by taking the $n$ most common verbs in subject dependency relations, testing different values for $n$. Table 5.4 shows these results. Interestingly, the classifier with only subject features performs best at a frequency cutoff of 25,000, which leaves only 86 verb features. Using a large amount of features, where the additional features are also more sparse (less frequent verbs) results in enough noise that the classifier cannot perform well, despite using feature weighting measures like Information Gain to decrease the weight of uninformative features.

Timbl also applies feature weighting, as explained in section 4.4.3. In short, this step determines which features contribute the most to successful classification decisions. Therefore, by looking at Information Gain feature weights, we can say which of the verbs used as features provide the most information about the animacy of their subject in our classifier. We have listed the 8 most informative verbs for the 50,000 frequency cutoff in table 5.5, which includes 39 verbs in total. Unfortunately, this metric does not show which class the verb is informative for. We can see the highest frequency verbs, have and be, which likely happens because they have the largest amount of different feature values,
<table>
<thead>
<tr>
<th>Verb Frequency cutoff</th>
<th>Number of verbs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0</td>
<td>77.30%</td>
</tr>
<tr>
<td>5000</td>
<td>360</td>
<td>84.50%</td>
</tr>
<tr>
<td>10000</td>
<td>201</td>
<td>84.24%</td>
</tr>
<tr>
<td>25000</td>
<td>86</td>
<td>91.14%</td>
</tr>
<tr>
<td>50000</td>
<td>39</td>
<td>90.72%</td>
</tr>
<tr>
<td>100000</td>
<td>17</td>
<td>88.39%</td>
</tr>
</tbody>
</table>

Table 5.4: Classifier performance for different numbers of subject features

<table>
<thead>
<tr>
<th>Verb</th>
<th>Translation</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>zie</td>
<td>to see</td>
<td>0.5795</td>
</tr>
<tr>
<td>heb</td>
<td>to have</td>
<td>0.5416</td>
</tr>
<tr>
<td>lig</td>
<td>to lie (down)</td>
<td>0.4240</td>
</tr>
<tr>
<td>verschijn</td>
<td>to appear</td>
<td>0.4130</td>
</tr>
<tr>
<td>sla</td>
<td>to stand</td>
<td>0.3847</td>
</tr>
<tr>
<td>ben</td>
<td>to be</td>
<td>0.3758</td>
</tr>
<tr>
<td>zeg</td>
<td>to say</td>
<td>0.3755</td>
</tr>
<tr>
<td>krijg</td>
<td>to get</td>
<td>0.3329</td>
</tr>
</tbody>
</table>

Table 5.5: The 8 most informative verbs (that are above the frequency cutoff) about the animacy of their subject, according to the Information Gain feature weighting value

simply due to being the most frequent. A weakness of Information Gain is that it overestimates the importance of features with a large number of different values. Still, we can see some interesting verbs as well, such as appear, a verb that might indicate inanimacy due to the passiveness of its semantics, or see, which implies some sort of sentience in its subject. However, this is just speculation, since we cannot quantify the verb's contribution to each class.

To confirm the effect of verb frequency, we have also checked this top-8 list for the classifier with the 10,000 verb frequency cutoff, which includes the 201 most frequent verbs, and therefore includes more verbs with lower frequency than the previous list. This list is shown in table 5.6. We can see that only 3 of the verbs are different from the previous list — come, find/think of, and seem has replaced the similar verb appear. All of these changes occurred near the bottom of the list, and the two most common verbs are in the top two spots now. The three new verbs are still fairly common ones, they just didn't make the cutoff before. The verbs see, lie and appear that appeared in the previous list, dropped to 14th, 21st and 70th respectively. It seems like lower-frequency verbs are indeed not very informative for our classifier for subject nouns, probably due to data sparseness. Less frequent verbs generally appear with fewer nouns that we would want to classify.
Table 5.6: The 8 most informative verbs about the animacy of their subject, with a larger feature set

<table>
<thead>
<tr>
<th>Verb</th>
<th>Translation</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>heb</td>
<td>to have</td>
<td>0.4166</td>
</tr>
<tr>
<td>ben</td>
<td>to be</td>
<td>0.4164</td>
</tr>
<tr>
<td>seq</td>
<td>to say</td>
<td>0.3942</td>
</tr>
<tr>
<td>krijg</td>
<td>to get</td>
<td>0.3655</td>
</tr>
<tr>
<td>kom</td>
<td>to come</td>
<td>0.3347</td>
</tr>
<tr>
<td>sla</td>
<td>to stand</td>
<td>0.3205</td>
</tr>
<tr>
<td>lijk</td>
<td>to seem</td>
<td>0.3195</td>
</tr>
<tr>
<td>vind</td>
<td>to find, to think of</td>
<td>0.3086</td>
</tr>
</tbody>
</table>

Table 5.7: Classifier performance for different numbers of object features

<table>
<thead>
<tr>
<th>Verb frequency cutoff</th>
<th>Number of verbs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0</td>
<td>80.92%</td>
</tr>
<tr>
<td>5000</td>
<td>455</td>
<td>83.14%</td>
</tr>
<tr>
<td>10000</td>
<td>242</td>
<td>85.68%</td>
</tr>
<tr>
<td>25000</td>
<td>79</td>
<td>90.95%</td>
</tr>
<tr>
<td>50000</td>
<td>39</td>
<td>92.68%</td>
</tr>
<tr>
<td>100000</td>
<td>18</td>
<td>92.93%</td>
</tr>
</tbody>
</table>

5.4 Verb object relations

We have evaluated the use of verb object features in the same way. Nouns in object positions are more likely to have the patient role, undergoing some sort of change, a role that inanimate entities often take. Again, we try to gain information on animacy from the selection restrictions of specific verbs, this time on the object role. For example, the verb to please prefers animate objects (Lamers and de Hoop, 2005):

(26) The holiday pleased the boy.
(27) ? The holiday pleased the box.

We have evaluated this feature type by taking the n most common verbs in object dependency relations, shown in table 5.7. Again, the classifier performs best with a low number of features, more so than the subject-based classifier. We can also see that its accuracy is higher in general. The 100,000 frequency cutoff with only 18 features results in the highest accuracy, but the 50,000 and 25,000 cutoffs also perform well.

We have also made a list of the 8 most informative verbs for object features in the best-performing classifier, shown in table 5.8. Again, we see see at the
Table 5.8: The 8 most informative verbs about the animacy of their object.

<table>
<thead>
<tr>
<th>Verb</th>
<th>Translation</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>zie</td>
<td>to see</td>
<td>0.5071</td>
</tr>
<tr>
<td>heb</td>
<td>to have</td>
<td>0.4821</td>
</tr>
<tr>
<td>noem</td>
<td>to call, to name</td>
<td>0.4049</td>
</tr>
<tr>
<td>maak</td>
<td>to make</td>
<td>0.4029</td>
</tr>
<tr>
<td>brng</td>
<td>to bring</td>
<td>0.3458</td>
</tr>
<tr>
<td>ken</td>
<td>to know</td>
<td>0.3343</td>
</tr>
<tr>
<td>speel</td>
<td>to play</td>
<td>0.2091</td>
</tr>
<tr>
<td>krijg</td>
<td>to get</td>
<td>0.1982</td>
</tr>
</tbody>
</table>

top of the list, but apart from the top 2 and get, the list features different verbs than the list for subject dependency features. That makes sense, because object selection restrictions are generally different than for subjects. Again, all the words are highly frequent, except play, though we can also see that there is a gap between the top 6 and the last two verbs. I do not have a clear intuition why play or the other verbs are so informative, but this classifier performs quite well at 92.93%, so there must be some clear probabilistic tendencies in object selection restrictions that the classifier can pick up on.

### 5.5 Adjective relations

Lastly, we have ran the same experiments for the dependencies of adjectives and their heads, in which the head is a noun that may be animate or not. Adjectives can be selective in what nouns they occur with as well:

(28) The quick brown fox jumps over the lazy dog.
(29) ? The quick brown fox jumps over the lazy box.
(30) He saw the full box.
(31) ? He saw the full dog.

In this case, lazy generally only takes nouns that are above some cutoff point in the animacy hierarchy — humans and animals can be used, but plants or objects don’t make sense, apart from some exceptional collocations such as lazy chair. The same goes for full, but with inanimate heads, though there are exceptions here as well.

We have again evaluated a classifier with only these adjective dependency features, the results of which are shown in table 5.9. As for the verb features, excluding the lower-frequency adjectives is quite beneficial. The best-performing system includes only 69 features in this case. Performance is slightly worse than for the verb-based dependency classifiers, but still well above the baseline.
<table>
<thead>
<tr>
<th>Adjective frequency cutoff</th>
<th>Number of verbs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0</td>
<td>77.69%</td>
</tr>
<tr>
<td>5000</td>
<td>747</td>
<td>78.27%</td>
</tr>
<tr>
<td>10000</td>
<td>284</td>
<td>85.49%</td>
</tr>
<tr>
<td>25000</td>
<td>146</td>
<td>86.61%</td>
</tr>
<tr>
<td>50000</td>
<td>69</td>
<td>88.99%</td>
</tr>
<tr>
<td>100000</td>
<td>29</td>
<td>84.97%</td>
</tr>
</tbody>
</table>

Table 5.9: Classifier performance for different numbers of adjective features

<table>
<thead>
<tr>
<th>Adjective</th>
<th>Translation</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>jong</td>
<td>young</td>
<td>0.5795</td>
</tr>
<tr>
<td>ander</td>
<td>other</td>
<td>0.5416</td>
</tr>
<tr>
<td>prachtig</td>
<td>wonderful</td>
<td>0.4240</td>
</tr>
<tr>
<td>Nederlands</td>
<td>Dutch</td>
<td>0.4130</td>
</tr>
<tr>
<td>verschil</td>
<td>different, various</td>
<td>0.3847</td>
</tr>
<tr>
<td>groot</td>
<td>great, large</td>
<td>0.3798</td>
</tr>
<tr>
<td>nieuw</td>
<td>new</td>
<td>0.3755</td>
</tr>
<tr>
<td>licht</td>
<td>light</td>
<td>0.3329</td>
</tr>
</tbody>
</table>

Table 5.10: The 8 most informative adjectives about the animacy of their head noun, according to the Information Gain feature weighting value

The list of the most informative adjectives for the 25,000 frequency cutoff is shown in table 5.10. This list is interesting, because none of these adjectives seem to have clear selection restrictions on animacy. Maybe young is more likely to occur with animate heads, though one can have young cheese as well. But judging by the accuracy scores obtained, it is enough for the classifier to work with.

The three feature types we have just described all perform fairly well on their own, but in the next chapter, we will evaluate a combined classifier that uses these feature types all at the same time.
Chapter 6

Evaluation

So far, we have shown for classification to be possible when using three feature types — subject, object and adjective dependency relations. Since all three feature types achieved decent classification accuracy by themselves, we used all three in our animacy classifier. In this chapter, we will discuss the combined classifier with all the features, evaluate it, and present some error analysis.

6.1 Features

In chapter 5, we have discussed our classification features, and determined the optimal number of features for each type of feature. Table 6.1 shows a summary of these results. The best accuracy is obtained by a classifier using a low number of object features. We thought that using these feature sets that performed best for each individual feature type to make a combined classifier would lead to the best performance. However, when we created a classifier with the features shown in table 6.1, i.e. the optimal ones from the previous chapter, the classifier accuracy was only 90.13%, worse than the subject or the object feature sets individually! This is unexpected, but it could be because the feature counts for the different types are unbalanced: only 18 object features to 155 others. We experimented by varying the cutoffs for each feature type, and found that balancing out the feature types indeed improved the combined classifier performance (table 6.2). Even though it does not have the absolute highest accuracy score, we decided to continue with the most balanced alternative of having a cutoff of 50,000 for each feature. The accuracy difference with the other two combinations is minor, and further experiments (line 5 and 6 of the table) showed that balance seems to be important. Performance dropped to around 86% by skewing the feature type ratios more.
<table>
<thead>
<tr>
<th>Feature type</th>
<th>Frequency cutoff</th>
<th>Number of verbs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject rels</td>
<td>25000</td>
<td>86</td>
<td>91.14%</td>
</tr>
<tr>
<td>Object rels</td>
<td>100000</td>
<td>18</td>
<td>92.93%</td>
</tr>
<tr>
<td>Adjective rels</td>
<td>50000</td>
<td>69</td>
<td>88.99%</td>
</tr>
</tbody>
</table>

Table 6.1: The best classifier results of each dependency feature type

<table>
<thead>
<tr>
<th>Cutoff-subj</th>
<th>Nr-subj</th>
<th>Cutoff-obj</th>
<th>Nr-obj</th>
<th>Cutoff-adj</th>
<th>Nr-adj</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>25000</td>
<td>86</td>
<td>100000</td>
<td>18</td>
<td>50000</td>
<td>69</td>
<td>90.13%</td>
</tr>
<tr>
<td>25000</td>
<td>86</td>
<td>50000</td>
<td>39</td>
<td>50000</td>
<td>69</td>
<td>92.68%</td>
</tr>
<tr>
<td>50000</td>
<td>39</td>
<td>50000</td>
<td>39</td>
<td>50000</td>
<td>69</td>
<td>92.76%</td>
</tr>
<tr>
<td>50000</td>
<td>39</td>
<td>100000</td>
<td>18</td>
<td>25000</td>
<td>146</td>
<td>86.02%</td>
</tr>
<tr>
<td>50000</td>
<td>39</td>
<td>50000</td>
<td>39</td>
<td>25000</td>
<td>146</td>
<td>85.44%</td>
</tr>
</tbody>
</table>

Table 6.2: Classifier results for different object/subject/adjective feature proportions

### 6.2 Best settings

In this section, we evaluate the classifier that we found to be optimal, based on the work described in this thesis so far. To summarize, we have a k-nearest neighbour classifier as implemented in Timbl. It is trained on noun dependency information gathered from the Lassy Large annotated corpus, combined with a noun animacy dictionary extracted from the Cornetto lexical-semantic database. The Timbl parameters (i.e. number of nearest neighbours, feature weighting) are automatically determined with the Wrapped Progressive Sampling algorithm for each trained classifier. The classifier features are the 39 most frequent verbs that occur in a subject relation with nouns from the animacy dictionary in the Lassy corpus, as well as the 39 most frequent verbs in such object relations, and the 69 most frequent adjectives in such adjective relations. The feature value metric used for these features is their Pointwise Mutual Information score with the noun. As our data set, we use all of the nouns from the animacy dictionary that occur at least 10 times in the Lassy corpus in a relevant dependency relation.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>80.92%</td>
<td>0.33</td>
</tr>
<tr>
<td>Best settings</td>
<td>92.27%</td>
<td>0.6495</td>
</tr>
</tbody>
</table>

Table 6.3: Performance of a classifier with optimal settings
We perform ten-fold cross validation to obtain an average accuracy score over ten runs (training and testing with different parts of the data set). Our baseline is a classifier that classifies every noun as the majority class \textit{inanimate}.

Table 6.3 shows the performance of this system. Along with the accuracy, we have also listed the F-score, which is the harmonic mean of the precision and recall. Precision, recall and F-score are commonly used metrics to describe the performance of a natural language processing system. These metrics are based on the following values:

- **True Positives**: The number of nouns that are class C, and are correctly predicted to have class C by the classifier.
- **False Positives**: The number of nouns that are not class C, but are incorrectly classified as class C.
- **True Negatives**: The number of nouns that are not class C, and are correctly classified as such.
- **False Negatives**: The number of nouns that are class C, but are incorrectly predicted to be of another class.

Given these definitions, precision represents the number of times the classifier was correct about a prediction for class C, and it is computed as follows:

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{6.1}
\]

Recall represents how often a noun of class C was correctly classified as such positive (P)

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{6.2}
\]

The F-score then represents these two measures in one measure. It is the harmonic mean of the two values:

\[
F - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{6.3}
\]

However, this measure only reports scores per class. A good way to summarize this is by taking their macro-average, which sums the F-scores for each class and divides it by the number of classes in the training set (3). Since we are using ten-fold cross-validation, we will still get 10 F-scores this way, but Timbl reports their average, as it does for accuracy (Daelemans et al., 2007). For the baseline, the F-score is 0 for the human and nonhuman classes, 1 for the inanimate class and therefore 0.33 overall.

An overall accuracy score of 92.5% still includes many mistakes, especially when the baseline is 80%. This is also indicated by the low macro-averaged F-score of 0.65, which indicates that one of the component F-scores (i.e. the score on one of the classes) is very poor. A major cause of errors is revealed when we examine the confusion matrix of this classifier, in table 6.4. As explained in section 4.5, a confusion matrix plots the class predicted by the classifier against their real class as stated in the animacy dictionary. It reveals that this classifier
has not learned the properties of the Nonhuman class, and rarely even attempt to classify any noun as nonhuman animate. It only does so 17 times, 14 of which are wrong, an accuracy of 18%. The Human class is predicted with 87% accuracy, and the large Inanimate class is predicted correctly 98% of the time. In section 4.2 we discussed the definition of the classes, and the Nonhuman class was indeed somewhat vaguely defined, including words that are likely to occur in animate contexts such as animals and groups of humans, as well as words that refer to entities that are only vaguely biologically animate, but are unlikely to be very agentic linguistically. This particular class may be problematic or even impossible to infer from the features that we use. The Cornetto lexical-semantic database is the only large animacy-annotated resource, so we are limited to using the categories that they defined, however. We did explore the idea of using a two-way classifier, which will be discussed in section 6.4. We also checked if the Nonhuman class is really impossible to learn by creating a balanced data set in which each class is equal in size (section 6.5). We will also discuss some additional ideas in chapter 7. But first, we will evaluate the classifier’s performance for data sets with different noun frequencies.

### 6.3 Noun frequency

In our experiments so far, we have simply chosen to use a dataset of all nouns with a frequency of 10 or more, since nouns with fewer instances than that will be very difficult to classify due to sparse data (such nouns only occur with up to 10 features at most). However, for some applications it may be interesting to examine other data sets as well. In the evaluation of their animacy classifier, Øvrelid (2009) included results for different accumulated frequency bins, i.e. all instances above a certain threshold of frequency. The motivation for this is the fact that low-frequency instances are more difficult to classify, there are fewer tokens available in the corpus to gain information from. They indeed observe better accuracy on data that is less sparse. These results are listed in table 6.5, but cannot be compared directly due to differences in the task that were explained in section 3.1. However, we expect to find the same pattern related to noun frequency in our data sets with different frequency cutoff values.

In our case, there is an additional problem that Øvrelid (2009) did not

<table>
<thead>
<tr>
<th>Confusion</th>
<th>Human</th>
<th>Nonhuman</th>
<th>Inanimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>153</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>Nonhuman</td>
<td>1</td>
<td>3</td>
<td>51</td>
</tr>
<tr>
<td>Inanimate</td>
<td>7</td>
<td>13</td>
<td>966</td>
</tr>
</tbody>
</table>

Table 6.4: Confusion matrix of the classifier with optimal settings. Columns are classes predicted by the classifier, and rows are actual noun classes. **Bold** values are correct predictions.
Table 6.5: Classifier performance of Øvrelid (2009) for different frequency bins

<table>
<thead>
<tr>
<th>Freq. bin</th>
<th>Instances</th>
<th>Baseline</th>
<th>MBL Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5,481</td>
<td>91.3%</td>
<td>93.9%</td>
</tr>
<tr>
<td>10</td>
<td>3,786</td>
<td>90.8%</td>
<td>95.4%</td>
</tr>
<tr>
<td>50</td>
<td>2,278</td>
<td>90.6%</td>
<td>96.1%</td>
</tr>
<tr>
<td>100</td>
<td>1,068</td>
<td>90.5%</td>
<td>96.8%</td>
</tr>
<tr>
<td>500</td>
<td>597</td>
<td>88.9%</td>
<td>97.3%</td>
</tr>
<tr>
<td>1000</td>
<td>291</td>
<td>80.3%</td>
<td>97.3%</td>
</tr>
</tbody>
</table>

have. Because we are using one resource to get our animacy dictionary and a different resource to get the feature values, it may be the case that words from the dictionary are unknown in the corpus (or at least do not occur in any relevant dependency relations), and thus no training or testing data can be obtained. These words thus cannot be defined in terms of dependency features, and therefore cannot be tested with at all. This is different from the problem of classifying an unknown word, which does not occur in the training data but shows up in the test data. In that case, features from the test data are available to base the classification decision on.

We are using a large corpus for feature extraction, so this situation should not occur very often in theory. However, the Cornetto database contains some rather unusual words, as well as proper names, which may cause this issue. In addition, the dictionary and the corpus are sometimes lemmatized differently, meaning that a word may occur in both resources, but in a different form. The most common case of this is compound nouns. For example, the word Tweede-Kamerfractie from the dictionary occurs as Tweede-Kamer_fractie in the corpus, and the word kerstdag is kerst_dag. However, not all compound nouns are split up with an underscore in this way. We have not performed any pre-processing to remedy this, though it should be possible using a compound splitter for Dutch, or a heuristic that identifies which underscores in the corpus represent compound splits.

For this reason, we have distinguished between a frequency cutoff of 0 and 1. The data set with the cutoff 0 includes all nouns in the animacy dictionary, even unclassifiable ones. With the cutoff of 1, these are excluded, but this data set will still contain words that occur only once. These words can serve as unknown words for testing purposes — if they are in the training set they won’t be in the test set, and vice versa. Furthermore, we have evaluated some higher frequency cutoffs. The results are listed in table 6.6. Surprisingly, the accuracy scores for different frequency cutoffs show a different pattern than the one reported by Øvrelid (2009). The scores are mostly the same across the different sets, apart from the 0-cutoff with words that are impossible to classify. Our classifier does not perform much worse on low-frequency words, indicating that it generalizes well. The drop-off for the highest frequency cutoff is interesting, it may be caused by a smaller data set (i.e. training + testing) in general. We apply
### Table 6.6: Classifier performance on datasets with different frequency cutoffs

<table>
<thead>
<tr>
<th>Frequency cutoff</th>
<th>Number of nouns</th>
<th>Baseline Accuracy</th>
<th>Classifier Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30.950</td>
<td>76.68%</td>
<td>84.14%</td>
</tr>
<tr>
<td>1</td>
<td>16.454</td>
<td>78.16%</td>
<td>90.82%</td>
</tr>
<tr>
<td>10</td>
<td>12.168</td>
<td>80.92%</td>
<td>92.27%</td>
</tr>
<tr>
<td>100</td>
<td>6.276</td>
<td>84.00%</td>
<td>91.06%</td>
</tr>
<tr>
<td>1000</td>
<td>1.671</td>
<td>88.99%</td>
<td>88.62%</td>
</tr>
</tbody>
</table>

The frequency cutoff to both the training data and the testing data, i.e. we are training a classifier for the task of classifying high-frequency words. In this, we follow the methodology of Øvrelid (2009). Her data set also becomes small, but their features are less complex, which might explain this difference.

### 6.4 Two-way classification

There is something to be said for an animacy classifier that decides only between animate and inanimate. After all, as we have seen in chapter 2, animacy effects in linguistics generally divide the hierarchy into two classes around some cutoff point, and if the noun is on the animate side, we use one construction, and on the inanimate side, we use the other (i.e. who and which). However, we are limited by the available animacy data from the Cornetto database. It divides the hierarchy into human and nonhuman animate, as well as inanimate. According to this scheme, human and nonhuman could together be considered an animate class, but we have seen that the nonhuman class is too broadly defined. Nevertheless, we will now attempt such an animate-inanimate classification task, as well as a two-way human-nonhuman classification task for comparison. In this case, the inanimate class was added to the nonhuman animate class to form a single class of everything that is not human. Apart from this merging of classes, we use the same setup as before.

Table 6.7 shows the results. We can see that the human-nonhuman classifier performs very well, even though the baseline is also high. The performance of the animate-inanimate classifier is comparable to our three-way classifier,
Table 6.8: Confusion matrices of two kinds of two-way animacy classifiers

<table>
<thead>
<tr>
<th>Confusion</th>
<th>Animate</th>
<th>Inanimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animate</td>
<td>157</td>
<td>73</td>
</tr>
<tr>
<td>Inanimate</td>
<td>0</td>
<td>986</td>
</tr>
</tbody>
</table>

(a) Animate-Inanimate

<table>
<thead>
<tr>
<th>Confusion</th>
<th>Human</th>
<th>Nonhuman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>154</td>
<td>21</td>
</tr>
<tr>
<td>Nonhuman</td>
<td>10</td>
<td>1031</td>
</tr>
</tbody>
</table>

(b) Human-Nonhuman

Table 6.9: Classifier performance on a data set where each noun animacy class is equally frequent (the classes are balanced).

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>39.91%</td>
</tr>
<tr>
<td>Balanced</td>
<td>72.14%</td>
</tr>
</tbody>
</table>

even though the task is easier. These results imply that the nonhuman class, even though it is described as animate, is closer to the inanimate class, at least as far as its verb and adjective dependencies are concerned.

It appears that a very high accuracy in animacy classification can be achieved if the relevant distinction is human-nonhuman. Whether this is useful depends on the application that the output will be used for. For a task like mention detection of persons, such a classifier would be useful, for example.

In table 6.8, we have also included the confusion matrices for these classifiers. There are no surprises there, the human-nonhuman classes are distinguished very well, while the animate-inanimate classes get confused a bit more often. All of the confusion occurs in the animate class, the inanimate class is predicted with perfect accuracy. This makes sense, since the troublesome nonhuman class was merged into the animate class, which therefore now indeed contains some words that should probably be considered to be inanimate.

### 6.5 Data balancing

To gain insight into the cause of the nonhuman classification problem, we performed another experiment. While the definition of the nonhuman class is vague, another problem is that this class is only a small minority of the data, only around 5% depending on the cutoffs used. That might be too little data proportionally. So, we created another data set, taking all 566 nonhuman nouns, and adding an equal number of random human nouns and random inanimate nouns. In this set, each class thus makes up 33% of the data (before frequency cutoffs) and no class is underrepresented. This also lowers the baseline significantly, since the majority class is now proportionally smaller.
The results in table 6.9 show that the classifier clearly outperforms the baseline, although the result is not great. A look at the confusion matrix (table 6.10) is more revealing. The classifier can correctly classify quite a few instances of each class (the values in bold). Accuracy on the human class is 69%, for inanimate it’s 80%, and for nonhuman it’s 65%. These accuracies are quite proportional to the size of each class after frequency cutoffs: 32% for human, 40% for inanimate, and 28% for nonhuman. This classifier is now able to distinguish all three classes, although at a reduced accuracy. This experiment shows that the nonhuman animate class can be identified from these features, the machine learning algorithm just does not handle the lack of instances of this class well.
Chapter 7

Applications and future work

In this chapter, we will discuss some potential applications of this classifier, as well as options for improvement. While animacy annotation has many uses, we have to be aware of the limitations of the annotation that this classifier can provide. For example, there is not much we can change about the definition of the three animacy classes, unless a different animacy resource with a different annotation scheme is created, or a method for reducing the classifier's resource requirements is developed. We also have to keep in mind that automatic classification is never perfect, and use of this classification data in another NLP system will introduce errors. That may or may not be a problem depending on the application. This classifier is in turn also based on automatically annotated data with the potential of containing errors (the Lassy corpus) but the fact that the data set is much larger than what could be obtained by manual annotation makes up for that. With these reservations in mind, we will discuss some applications of the classifier.

7.1 Corpus annotation

As we clearly noticed when looking for resources to build an animacy classifier with, the amount of animacy-annotated resources available is quite limited, especially for Dutch. Our animacy classifier could be used to annotate more resources with animacy information, in a less labour-intensive way than manual annotation.

If the corpus is to be used for linguistic research, a higher level of accuracy than what our classifier has achieved would probably be desired. This can be done by resorting to semi-automatic annotation. In this procedure, animacy annotation is first generated automatically, and is then checked by an expert. This is more efficient than having an expert annotate everything from scratch, but still involves a significant manual effort. Our three-class animacy hierarchy would also be a limiting factor in this case. As we have seen in section 2.4, semantically annotated corpora that are used for linguistic research use more fine-grained animacy hierarchies than that. In particular, they involve large
hierarchies of animacy, which our classifier probably could not learn from the features that we used, although we have not tested this. And as mentioned, we cannot even change the definition of the somewhat vague class of NONHUMAN without the availability of another resource. In addition, since our classifier is lemma- or type-based, it cannot take local context into account if it would be used to annotate a text. This would be very difficult to do automatically (section 4.1), though in the case of semi-automatic annotation, such errors would be noticed by the expert annotator.

If the corpus is to be used for statistical natural language processing tasks, the error rate may not be such a problem, as was the case for the Lassy corpus when we trained our classifier on it. However, there cannot be structural errors, since they would skew the statistical model too much. An example of such an error is the inability of our classifier to learn about the NONHUMAN class, which is classified with 0% accuracy in some cases. For this kind of application, one would have to use the classifier trained on balanced data (section 6.5), or a two-class version that avoids this issue (section 6.4). The NONHUMAN class may not be useful for some applications, so this could be an acceptable compromise in those cases.

7.2 Languages and resource dependence

It should be noted that potential applications are not necessarily limited to the Dutch language. Like most statistical methods, the method we use is fairly language-independent. It should work for any language in which there are some sort of dependency relations that place semantic selection restrictions on nouns. The main issue is to find the right resources that can be used for training. For languages that are rich in resources, such as English, this should not be a problem, though for most languages of the world, there are no readily available lexical-semantic databases.

Our method requires some sort of dictionary with animacy information in it, which could be extracted from a lexical-semantic database, for example. It also requires a dependency-parsed corpus with labeled roles, and it requires both of the resources to have undergone some sort of lemmatization. The method can be simplified if a corpus with word-sense or semantic annotation that includes animacy information is available. In this case, the animacy dictionary is not needed, and the corpus can be used directly.

Those are still fairly strict resource requirements, and it would be interesting to try a semi-supervised method. This would reduce the resource requirements, making it easier to use the method for different languages or domains. A method that is sometimes used in semantic classification tasks involves using a small amount of known examples (i.e. known animates and inanimates), and automatically expanding that list. An example is the work by Hu and Liu (2004) on opinion mining, in which adjectives are classified as positive or negative. They start with a small, manually crafted list of adjectives, and expand it using WordNet. Such a technique may be worth trying for animacy annotation as well, where expansion could take place through WordNet as discussed
by Orasan and Evans (2007), or through an iterative process of self-training, as in the method of de Ilarraza et al. (2002) discussed in section 3.3. In such a process, the classifier is trained on a small seed set of frequent nouns. These frequent nouns should occur in a lot of different dependencies, providing knowledge about the animacy selection restrictions of those dependencies for the classifier to use. The output of this classifier would then be used for training itself, a process that can be repeated, in the hope that it generalizes to other, less frequent nouns. But like any automatic natural language processing, this may introduce too many errors.

7.3 NLP tasks

In the related work that we discussed in chapter 3, examples of NLP tasks for which animacy classification can be used were discussed. Øvrelid (2009) proposed to incorporate animacy information into a parser, and indeed evaluated whether the output of her own animacy classifier would contribute to better parsing accuracy for the MALT dependency parser. She reports a small but significant improvement in the Labeled Attachment Score of the parser. One way in which animacy may aid parsing is during disambiguation. Imagine a situation where there are two alternative parses involving a known animate noun, in one parse it is the object and in the other parse, it is the subject. Animate nouns are far more likely to be the subject in most cases. There are also several grammatical phenomena where animacy plays a role, though this is not very prominent in Dutch, as discussed in section 2.1.

de Ilarraza et al. (2002) were motivated to perform animacy classification by ambiguity problems in machine translation to the Basque language, where a common preposition is ambiguous when translated from Spanish, and the animacy property of the head is needed for disambiguation. This is a specific situation, but similar ambiguities exist in Dutch, for example when translating the Dutch word *die* in this sentence:

\[(32)\]  De *man* **die** op *de* *tafel* zat.
The man **who** on the table sat.

`The man **who** sat on the table.'

We need to know that *man* is animate in order to use **who**, the correct English translation of *die* in this case. If the subject had been inanimate, we would have needed **which**. Another option is **that**, and furthermore the Dutch word could also translate to **those** and **these** in different contexts.

Orasan and Evans (2007) developed their animacy classifier to aid in anaphora resolution. If the animacy of nouns is known, it becomes easier to decide what pronouns like **she** and **it** refer to. The pronoun **she** should only be resolved to a human noun. This same problem exists in Dutch, except that **hij** (he) and **ze/zij** (she) can refer to animals as well as humans — the cutoff point is lower on the animacy hierarchy. As we have seen, our classifier has some problems with these borderline cases, but it should at least be able to make a helpful contribution in resolving **hij** and **ze/zij** to human referents.
7.4 Classifier improvement

We have already noted that the classifier could be improved by reducing its dependence on linguistic resources. Somewhat related to this is the possibility of doing preprocessing to make optimal use of the available data. Since such preprocessing generally involves language-specific rules, we have refrained from it, but it could be a good way to improve the classifier for Dutch. An example of this was described by Baker and Brew (2010) for Japanese. They made use of the compound morphology of Chinese loanwords in Japanese to make the classification task easier. They grouped them by their suffix: for example, *firemen, salesmen* and *weathermen* are all types of men. They then consider these suffix groups as one item when doing feature extraction, obtaining a wider range of dependency relations than one would get for each of those words individually. While this probably wouldn't work as a general rule in Dutch or English, it might still be interesting to try some compound splitting heuristics for preprocessing. The current system already has a problem with compounds anyway, since the two resources that we use sometimes represent them differently.

Recently, another possibility for improvement has appeared. A word-sense annotated corpus of Dutch is being developed (Vossen et al., 2012), which creates the possibility of token-based animacy classification, as opposed to the type- or lemma-based classification that we have discussed. As far as we are aware, no such classifier exists for any language at the time of writing. In this corpus, each individual token is enriched with semantic annotation about its sense, including animacy information. And when this sort of training data is available, a classifier could be trained on it, using the features that we use. However, we already checked in this corpus whether token-based classification would actually solve many errors, and it did not seem to be the case. Out of 2072 sense-annotated nouns that also occur in the Cornetto lexical-semantic database, only 34 (1.5%) were ambiguous in terms of animacy, a situation that only token-based classification can resolve. That is a surprisingly small number, and thus it might not be worth doing token-based animacy classification.
Chapter 8

Conclusion

In this thesis, we have presented the first animacy classifier for Dutch. We have discussed the complexities of animacy and shown various examples of its effects in different languages. We have discussed animacy hierarchies and annotation schemes, the uses of animacy in natural language processing, as well as existing work on animacy classification for other languages. We then described our own approach to animacy classification, adapted to the state of the art of Dutch linguistic resources, and evaluated the resulting classifier system.

We used a k-nearest neighbour classifier as implemented in Timbl to classify nouns into three classes of animacy — **HUMAN**, **NONHUMAN ANIMATE** and **INANIMATE**, as defined by the annotation scheme of the Cornetto lexical-semantic database. Due to the availability of the Lassy Large corpus for Dutch, a large, automatically annotated corpus we were able to use novel kinds of distributional lexical features, i.e. dependency relations with specific words, to classify the nouns into animacy classes. A more basic approach of using syntactic distributional features did not yield good results. Words can impose semantic selection restrictions on their dependencies, for example, the verb *to think* mostly takes animate subjects. By quantifying such subject, object and adjective dependency relations of words for which the animacy is known, the classifier can gain information about such restrictions or tendencies related to animacy classes, and generalize this to novel words to predict their animacy class. We determine whether dependency relations are significant through measures of statistical association, the same method that is used for finding word collocations in general.

We have demonstrated a classifier accuracy of 92-93% on the task summarized above. However, this classifier has problems learning to distinguish the **NONHUMAN** category, so we have also explored some methods of avoiding that issue — using a different mix of training data, or simplifying the task to two-class classification. In classifying with a two-class scheme of **HUMAN**-**NONHUMAN** (other), we reached an accuracy of 97-98%. Lastly, we discussed some potential improvements and applications of the method, such as corpus annotation or use in other NLP tools.
Bibliography


