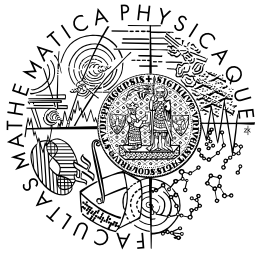


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Michal Auersperger

English grammar checker and corrector: the determiners

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: RNDr. Pavel Pecina, Ph.D.

Study programme: Computer Science

Study branch: Mathematical Linguistics

Prague 2017

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: English grammar checker and corrector: the determiners

Author: Michal Auersperger

institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Pavel Pecina, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Correction of the articles in English texts is approached as an article generation task, i.e. each noun phrase is assigned with a class corresponding to the definite, indefinite or zero article. Supervised machine learning methods are used to first replicate and then improve upon the best reported result in the literature known to the author. By feature engineering and a different choice of the learning method, about 34% drop in error is achieved. The resulting model is further compared to the performance of expert annotators. Although the comparison is not straightforward due to the differences in the data, the results indicate the performance of the trained model is comparable to the human-level performance when measured on the in-domain data. On the other hand, the model does not generalize well to different types of data. Using a large-scale language model to predict an article (or no article) for each word of the text has not proved successful.

Keywords: English determiners grammar checker

Contents

Introduction

The use of the indefinite and definite articles (*a, an, the*) in English poses a significant problem for many speakers with a different first language. English grammars analyze the use of the articles in great detail but they acknowledge the rules can be sometimes contradictory (e.g. conflict between the first mention (*an*) and cataphora (*the*) in the sentence: *I had the / an impression that something was wrong.* [Dušková et al., 2006, p.81]). As is often the case with language, grammars provide analyses of different tendencies effecting the language phenomenon rather than a fixed set of rules to follow. The aim of the thesis is to create an automatic grammar checker targeted specifically at the use of *the, a* and *an*.

Previous work and problem formalization

Considering the literature, the interest in article correction seems to date back to the middle of the nineties. In this section, a brief overview of the previous work will be provided. Also, some basic decisions that were made in the beginning of the project as well as their justification will be given.

In order to improve the quality of machine translation generated text, Knight and Chander [1994] used decision trees to assign noun phrases with *the* or *a/an* article with the accuracy of 78%. Minnen et al. [2000] used memory-based learning to predict definite, indefinite or zero article for noun phrases in Penn Treebank. Their accuracy is 83.6%. In the same task and with the same data, Lee [2004] achieves the accuracy of 87.7%. Turner and Charniak [2007] use language model trained on more data to achieve the accuracy of 86.6% on Penn Treebank corpus. More recently, grammatical error correction, including articles, was the focus of the shared task at two successive CoNLL conferences [Ng et al., 2013] and [Ng et al., 2014]. The systems of the participants were trained and tested on essays written by Singaporean students. However, article errors were not their only target. Sun et al. [2015] use convolutional neural networks on the same data.

Most of the reviewed authors [Knight and Chander, 1994], [Minnen et al., 2000], [Lee, 2004], [Turner and Charniak, 2007] choose first to identify noun phrases in the text and then consider the use of an article with respect to each of the noun phrase. Another approach, advocated by Sun et al. [2015], is to consider each possible position in the text to be a potential candidate for an article placement. The former approach stems from the fact that articles occur only with noun phrases and that their position within noun phrases is easy to predict (i.e. mostly the very beginning of the noun phrase). Moreover, not being

limited by an n-gram window provides more freedom in feature engineering. On the other hand, Sun et al. [2015] advocate the use of their approach by potential "propagation of errors in the existing tools of NLP" such as taggers and parsers that are necessary for extracting the noun phrases from the raw text. To facilitate comparison with what seems as larger body of literature on the subject, the former approach was selected for most of the experiments. However, a language model is also trained and evaluated for comparison.

As mentioned above, some researchers decided to train their models on texts written by students of English, others relied on standard corpora, i.e. texts written by English native speakers. Intuitively, the former approach makes more sense for practical grammar correction systems. However for this project, the latter option was chosen for the following reasons:

a) The student corpora are less accessible and contain less data.

b) Texts written by non-native speakers contain other kinds of errors that can influence performance of some preprocessing steps (tagging, parsing). Avoiding these types of errors makes it possible to focus solely on the articles.

c) Utilizing the articles already present in the input text seems practical (one could, for example, make more corrections in certain contexts while staying conservative in others). However, this means the resulting system would model not only the nature of the grammatical phenomenon, but also some systematic linguistic behavior in non-native speakers of English. This is problematic for two reasons. Firstly, there is not much consensus on the regularity of such behavior. Gressang [2010] contrasts different studies finding in turn that the single main source of article errors is omission, predominant use of *the* or predominant use of *a/an*. Rozovskaya and Roth [2010] show large variability in article error types based on the first language of the speaker. Secondly, the researchers who make use the original articles in text sometimes find they need to introduce artificial errors since the true errors are sparse [Rozovskaya et al., 2013], [Lee, 2004].

English articles come in three forms: *the*, *a* and *an*. In accord with previous research, in this thesis the two forms of the indefinite article are treated as one entity. The choice between *a* and *an* is a matter of simple phonological rules rather than a question of noun phrase determination. As such, the proper form can be chosen in a post-processing step.

Summing up the above, in this project, the article correction is understood as a classification task. Each noun phrase in the input text is assigned with one of

the three categories irrespective of the potential original article: definite article (*the*), indefinite article(*a/an*) and zero article (no surface realization).

Thesis structure

Chapter 1 provides linguistic perspective on the use of articles and determiners in general. In Chapter 2, two machine learning algorithms used for the experiments in this thesis are described, namely logistic regression and gradient boosted trees. The motivation for this choice of the methods is also mentioned. Chapter 3 deals with the experimental setup; namely, the processing of the data and the features used for the experiments. The evaluation of different approaches as well as comparison to human performance is given in Chapter 4.

1. English Articles from the Linguistic Perspective

The following chapter provides an introduction to the use of articles from the point of view of English grammars. In particular, the information is based on two established sources: very thorough grammar description by Quirk et al. [1985] and more theory-based work by Dušková et al. [2006].

Generally, there are four specific forms of articles in English: two forms of the indefinite article: *a* [ə], *an* [ən] and two forms of the definite article, whose distinction is, however, apparent only in the spoken language, – *the* [ðə] and *the* [ðiː]. For both the definite and indefinite articles, the longer variants, i.e. *an* and *the* [ðiː], occur in front of a (pronounced) vowel.[1]

The articles are one way of expressing the grammatical category of *definiteness*, which is a property of English noun phrases. Similarly to countability, another such grammatical category, it does not have a counterpart in the system of the Czech (and generally Slavic) noun. Presumably, this is why the correct use of English articles poses a large problem for Czech speakers.

From the semantic perspective, definiteness conveys the information about the nature of the reference of the noun. In other words, it helps distinguish different types of relationship between a noun phrase and the entity being referred to. This notion can be illustrated by the etymology of the two basic forms: the definite article originates from the old English demonstrative 'θœt' (that) and the indefinite article from the Old English numeral 'ān' (one). This closeness can be observed in some contemporary uses such as *for the moment (≃ for* this *moment)*, *Don't say a word (≃ not a* single *word).* [Dušková et al., 2006, p. 59] In greater detail, the relationship between definiteness and the reference of the noun will be discussed in Section 1.3.

1.1 Determiners and their position within the noun phrase

Generally, the grammatical category of definiteness is expressed by a closed-class group of words known as *determiners*. The articles are particular members of this group, other examples are demonstratives, possessive pronouns, cardinal and

[1] Not surprisingly, there are exceptions to this rule. Notably, the longer forms can be used for emphatic reasons even when not followed by a vowel sound: *He is **the** [ðiː] man!*

ordinal numerals and others.

Typically, all determiners occur at the beginning of the corresponding noun phrase. However, more than one determiner can be used simultaneously. With respect to the possible co-occurrence of different sets of determiners, Quirk et al. [1985, p. 253] describe the following small system of three determiner classes:

predeterminers *all, both, half, twice* ...

central determiners *a, an, the*, demonstratives, ...

postdeterminers cardinal and ordinal numerals, *many, few*, ...

When more determiners are used within a given noun phrase, they cannot be from the same category and they must follow the provided order of the categories. Thus, the following example *all the five boys* is valid because it contains a premodifier, a central modifier and a postmodifier in the given order; while the example *the five all boys* is invalid because the order is broken. [Quirk et al., 1985, p. 253] In practice, noun phrases often occur with only one (if any) determiner.

Apart from a possible preceding predeterminer (*once a week, twice as much, half the money*), Dušková et al. [2006, p. 60] give the following situations when an article is moved from its typical position at the beginning of the noun phrase:

- **modification of a pre-modifier by *so, as, too, however***
  This is [as good ***a*** *hotel as any other]. It's [too good a chance] to be missed*
- **any/no worse, no less**
  He is no worse a doctor for being ...
- **rather, quite**
  rather an unexpected result, quite a long time
- **exclamative sentences with *what***
  What a silly lie!

1.2 Countability

The usage of the definite and indefinite articles as well as other determiners is heavily influenced by another grammatical category of the English noun – countability. Semantically, countability reflects the distinction between the referred entities along the lines of continuous – discrete. In other words, countable nouns refer to classes whose instances are conceptually distinguishable from one another, while uncountable nouns (also called mass nouns) refer to homogeneous entities with no distinct units.

Understanding what is countable and what is not differs from language to language in certain situations. The same is true for the extent to which this distinction is manifested in the grammar of a language. Dušková et al. [2006, p. 50] contrast English with Czech, where the notion of countability is manifested only in the presence/absence of plurals, e.g. *vzduch* — **vzduchy* vs. *pes* — *psi*. On the other hand, in English the countability of nouns further influences the choice of other co-occurring words within the sentence, most notably the determiners. This is what makes countability a grammatical category.

1.3 Reference

As mentioned earlier, the grammatical category of definiteness (and its realization in the form of articles and other determiners) carries information about the type of reference of the noun phrase. This section deals with the relationship between different reference types and the corresponding types of determination.

The two main reference categories are the generic and specific reference. The first one is represented by such sentences as: *Men are from Mars, women are from Venus.* Here, both *men* and *women* refer to all the members of the given class. In contrast, the *stranger* in *Then a stranger came in* refers to a particular man, a specific instance of the class of all strangers. Therefore it expresses the specific reference. This type can be further divided into the definite and indefinite subtypes. The indefinite one is illustrated by the previous example. The *stranger* as the member of the class of strangers is not uniquely defined in the situation. The definite specific reference is illustrated by the sentence *I want to see my doctor*, where the *doctor* "is referring to something which can be identified uniquely in the contextual or general knowledge shared by speaker and hearer."[Quirk et al., 1985, p. 265] Table 1.1 shows the means by which these types of reference are expressed for both countable and uncountable nouns.

1.3.1 Generic reference

Generic reference is the least constrained type of reference. It can be expressed by a noun in plural form without any article (a), by a singular with either the definite (b) or the indefinite (c) article. Uncountable nouns can occur only without any article (d). All these options are illustrated by the following example taken from [Quirk et al., 1985, p. 281]:

(a) Bull terriers make excellent watchdogs.

(b) The bull terrier makes an excellent watchdog.

reference type		Countable	Uncountable
Generic		the cat a cat cats	music milk
Specific	Definite	the cat the cats	the music the milk
	Indefinite	a cat cats some cats	music, some music milk, some milk

Table 1.1: Means of expressing different types of reference for countable and uncountable nouns. The table is adopted from [Dušková et al., 2006, pp. 61–2]

(c) A bull terrier makes an excellent watchdog.

(d) Velvet makes excellent curtain material.

However, not all the variants are equally likely in each context. The most constrained variant is the indefinite article. For example, the noun phrase in the object position of the sentence *Nora has been studying a medieval history play* does not have generic interpretation. [Quirk et al., 1985, p. 281] Another factor limiting the use of the indefinite article for generic reference is the semantic class of the predicate. Dušková et al. [2006, p. 63] enumerate the following verbs with which the generic *a* does not occur: *abound, be rare, increase, decrease, ...*.

1.3.2 Specific definite reference

The fact that in the given situation, the referent is interpreted as definite can be based on many different sources of information. Both grammars, [Quirk et al., 1985] and [Dušková et al., 2006], illustrate this by listing many subtypes of the specific definite reference. While sometimes they differ in terminology and/or systematization of particular examples, the overall picture is largely consistent and is briefly provided here.

The first factor that can make the referent definite is the situation – **situational reference**. This type of reference is determined by the current extralinguistic situation, be it an immediate situation such as in *Pass me the paper, please*; where the object is directly visible (in such cases, the definite article can be changed for demonstrative *that*); or a more general setting, in which the noun is still uniquely identified: *Open the door, please. Did you hear the bell?*; or more general *We used to walk down to the river*; or even more general *Look, the Sun is rising.*

Other factors can be inter-textual. The **anaphoric reference** refers to situations where the referent of the noun is defined based on the previous discourse. This can be based on direct correspondence between two noun phrases: *We went to see a play, . . . , the play was not very good*; or more free correspondence: *I went down to pet her dog, but the beast almost bit me.* Sometimes the relationship between the noun and its antecedent is less straightforward and can be based on association: *I saw there an old guitar. I wanted to play it but found out the strings were broken.* Here the definite reference of the *strings* arises from its association to the *guitar*. (This situation is called associative anaphoric reference in [Dušková et al., 2006]).

The **cataphoric reference** is similar, but the noun is specified by what follows rather than what precedes it in the discourse. Nut surprisingly, the distance between the noun phrase and the part of the text that makes it definite is much more limited than in the case of the anaphoric reference. Mostly, it is expressed by different kinds of post-modification within the same sentence. Based on the syntactic properties of the post-modification, Dušková et al. [2006, pp. 67–9] list the following options for the catphoric reference (the examples are taken from the same source):

a) Postmodification by a restrictive relative clause
   This is illustrated by such examples as *I appreciate the initiative that you have all shown; Paul left with the girl who had come with George.* However, not all relative restrictive clauses signal the definite reference as in some cases, there still might be more than one potential referent left: *We took a train that stops at every station.* Further, some clauses may function as only qualifying the given noun phrase: *She carried a bag that was too large for her.* Such examples can sometimes be restated with a premodifier: *She carried a large bag.*

   Similarly, the cataphoric reference can be expressed by restrictive use of the infinitive *The man to deal with this matter is Bill*; participle *the examples given above*; or a prepositional phrase *I saw it in the shop at the corner of High street.*

b) Postmodification by an *of*-phrase
   Similarly to the previous case, postmodification by an *of*-phrase expresses the cataphoric reference if it is specific enough to identify a single member of the class denoted by the head of the noun phrase: *the roof of the house, the bottom of the sea.* This is not necessarily the case for all the *of*-phrases: *a page of the book.*

c) Content relative clause

The content clause refers to the same referent as its parent noun phrase and is used to make the referent more specific. Often, this results in the use of the definite article: *The fact that he has won several tournaments won't help him to pass his exams.* However, sometimes the content clause can be used for qualifying the noun phrase in which case *the* is not used: *Her father died at a time when she was too young to fully feel his loss.*

d) Apposition
Similarly to content relative clauses, apposition serves as a means of providing additional information about the referent. Often, this additional information leads to the fact the referent is uniquely defined: *the number seven, the river Thames, the City of London.*

Finally, Quirk et al. [1985] mention two other types of specific definite reference: **sporadic** and **logical**. The first type includes specific nouns that often occur with the definite article, such as *the radio, the Internet, the theater, the news, the train, the bus, ...*. What holds these examples together is the fact that in these cases the "reference is made to an institution which may be observed recurrently at various places and times"[Quirk et al., 1985, p. 269]. The other — perhaps more straightforward — category consists of examples where the uniqueness of the referent is inherently encoded in the meaning of a special modifier: *the next/previous/best/worse book, the only question I have, the final stop, ...*.

1.3.3 Specific indefinite reference

The specific indefinite reference is expressed in situations when there is only one referent but it is not uniquely defined. The most common example of this is the situation of the first notion, i.e. introducing a concept for the first time in the discourse: *Yesterday, I met a good friend of mine.* However, even if the referent is already implied in the discourse, indefinite article is sometimes used because of its non-uniqueness: *Take a seat.*

With this type of reference, the distinction between countable and uncountable nouns is expressed the most. Instead of the indefinite article, mass nouns take either (unstressed) *some* or they mark the indefinite reference by absence of a determiner (in contrast to the definite reference, where *the* is obligatory): *Would you like some chocolate? It takes courage to oppose him.*

1.4 Exceptions and conflicting tendencies in article usage

Unfortunately, it is not possible to give a complete system of the use of English determiners as there are always many exceptions to the general tendencies provided above. One such example is the class of proper nouns. Due to their semantic nature, they express specific definite reference, but unlike the common nouns, they often occur without a definite article: *Agatha Christie, little Emily, Europe, Norway, Oxford Street, Easter Monday, ...*; others are used with *the*: *the Burtons, the United States, the Crimea, the Congo, the Alps, the Shetlands, the City of New York, the Pacific Ocean, the Thames, the Times, ...* Furthermore, some proper nouns can become common nouns depending on the context: *A Mr. March to see you; He is not a Mozart;* while others can take an article depending on the type of modification *He wrapped the trembling Emily in his coat; the London I am talking about; the vision of a new Canada.* [Dušková et al., 2006, pp. 75–9]

Sometimes even common nouns do not occur with articles: *go to school, in court, go to bed, travel by train, go by car, from head to foot, hand in hand, in contrast with, in support of, in fear, in trouble, take office, give way, ...* [Dušková et al., 2006, pp. 79–81]

As mentioned in the introduction, there are cases where there are more options with respect to the choice of articles. This can result from different interpretations of the situation: *I discussed an interesting project with Jim last night. Afterwards I went to discuss (a/the) project with Fred.* [Dušková et al., 2006, p. 66] Similarly, the use of an article can differ with respect to the countable/uncountable interpretation of the noun: *There was a short silence — There was absolute silence.* Yet in other cases, the options reflect virtually no modification in the meaning: *throw (a) new light on, at this time of (the) day, in (the) summer, take (a) pride in sth., watch (the) TV ....* [Dušková et al., 2006, pp. 81–2]

2. Machine Learning Algorithms

In the following chapter, two machine learning algorithms used for the experiments in this thesis are described, namely logistic regression and gradient boosted trees.

The motivation for this choice is mainly the ability of both algorithms to process sparse data: logistic regression (in its form of the maximum entropy model) is an established approach for many problems in the area of natural language processing, where sparsity is inherent to most of them (for a review of maximum entropy model use cases in NLP, see [Ratnaparkhi, 1998]); the other algorithm — gradient boosted trees — has recently seen a new implementation that can handle sparse data efficiently [Chen and Guestrin, 2016].

Moreover, logistic regression was used by Lee [2004], whose experiment is replicated as part of this thesis (for motivation see Chapter 3). Another reason for selecting gradient boosted trees is the fact that its implementation — XGBoost — reportedly achieves very good results on many different tasks (as demonstrated by Kaggle competitions) [Chen and Guestrin, 2016], while general gradient boosted trees have proven effective in large scale production code [He et al., 2014].

2.1 Logistic regression

2.1.1 Binary logistic regression

In the strict sense, logistic regression learns a hypothesis function in the following form:

$$h_\theta(x) = g(\theta^\mathsf{T} x), \tag{2.1}$$

where x is a vector of features representing the classified example,[1] θ is a vector of weights assigned to the features and g is the logistic function:

$$g(z) = \frac{1}{1 + e^{-z}}. \tag{2.2}$$

The hypothesis represents the probability that the given example is positive, which makes it suitable for binary classification only: $h(x) = p(y = 1|x)$.

Training

The problem of estimating the parameters of the model (the θ vector) is — in the paradigm of supervised learning — formulated in terms of minimizing a cost

[1] In order to simplify the notation, for m features, x is an (m+1)-dimensional vector with the additional element $x_0 = 1$, thus $\theta^\mathsf{T} x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m$.

function, i.e. a function that measures the error a model is making. In the case of logistic regression the cost can be expressed as

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(h_\theta(x_i)) + (1-y_i)\log(1-h_\theta(x_i))] + \frac{1}{m}\lambda\sum_{j=1}^{n}\theta_j^2. \quad (2.3)$$

Here, m stands for the number of training examples, n stands for the number of features, x_i and y_i represent the feature vector and the true label of the i^{th} example. The last term $\frac{1}{m}\lambda\sum_{j=1}^{n}\theta_j^2$ is a regularization term that controls the complexity of the model. The strength of the regularization can be tuned by the parameter λ.

By minimizing the cost function of the given form, one maximizes the likelihood $\mathcal{L}(\theta|x)$, which is defined as the probability of the observed data given the model:

$$\mathcal{L}(\theta|x) = P(x|\theta) = \prod_{i=1}^{m} p_\theta(y_i|x_i) = \prod_{i=1}^{m} h_\theta(x)^{y_i}(1-h_\theta(x))^{1-y_i}. \quad (2.4)$$

Now, the cost function given in (2.3) is equal to $-\frac{1}{m}\log\mathcal{L}(\theta|x)$ plus the regularization term.

For minimizing the cost function, gradient descent can be used, however in practice, more efficient algorithms are usually chosen. The implementation of logistic regression used for this thesis uses the so called LBFGS algorithm [Nocedal, 1980]. It is based on the Newton's method, a procedure that iteratively approaches a local minimum of a function[2] by setting

$$x_t := x_{t-1} - \alpha\frac{f'(x_{t-1})}{f''(x_{t-1})}, \quad (2.5)$$

where x_t and x_{t-1} are the values of x at the step t and $t-1$, respectively; and $\alpha \in (0,1)$ is a learning rate, i.e. the parameter determining the speed of convergence. This update rule comes from approximating the function f by its second order Taylor expansion at the point x_{t-1}:[3]

$$f(x_t) = f(x_{t-1}+\Delta x) \simeq f(x_{t-1}) + f(x_{t-1})'\Delta x + \frac{1}{2}f(x_{t-1})''\Delta x^2. \quad (2.6)$$

Setting the first derivative of the approximation of the function with respect to

[2]Since the cost function in (2.3) is convex, the algorithm finds the global minimum of the function.

[3]Taylor series approximates a function $f(x)$ at a point a by $\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a^{(n)})^n$, where $f^{(n)}(a)$ is the n^{th} derivative of f evaluated at a.

Δx to be zero and solving for Δx, the update rule in (2.5) is derived as:

$$\begin{aligned} 0 &= \frac{d}{d\Delta x}\left[f(x_{t-1}) + f(x_{t-1})'\Delta x + \frac{1}{2}f(x_{t-1})''\Delta x^2\right] \\ &= f(x_{t-1})' + f(x_{t-1})''\Delta x \\ \Delta x &= -\frac{f'(x_{t-1})}{f''(x_{t-1})} \end{aligned} \tag{2.7}$$

For multiple dimensions, the update rule (2.5) changes into:

$$x_t = x_{t-1} - \alpha[Hf(x_{t-1})]^{-1}\nabla f(x_{t-1}), \tag{2.8}$$

where x_t and x_{t-1} are vectors. $\nabla f(x_{t-1})$ and $Hf(x_{t-1})$ represent the gradient and the Hessian matrix of the function and, given the function f takes m parameters $x_1, x_2, \ldots x_m$, are defined as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}$$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_m x_1} & \frac{\partial^2 f}{\partial x_m x_2} & \cdots & \frac{\partial^2 f}{\partial x_m^2} \end{bmatrix}.$$

Since computing the inverse of the Hessian matrix can be computationally very expensive, the LBFGS algorithm approximates it by monitoring the history of the updates. Specifically, it stores the information on the change of x and the corresponding change of the gradient over a fixed amount of steps. Based on this information, the update step is approximated by first iteratively estimating the inverse Hessian matrix and the first 'version' of the update direction $[Hf(x_{t-1})]^{-1}\nabla f(x_{t-1})$, then this update direction is iteratively improved based on the same information from the history.

2.1.2 Maximum entropy model

Ratnaparkhi [1998] favors the use of the maximum entropy framework over logistic regression for NLP tasks. This algorithm learns the probabilities of an example belonging to each class from a set of classes and thus can be used directly for

multinomial classification:

$$p(y|x) = \frac{\prod_{j=1}^{k} \alpha_j^{f_j(y,x)}}{\sum_{y'} \prod_{j=1}^{k} \alpha_j^{f_j(y',x)}}, \tag{2.9}$$

y is the target class, x is the example or the 'context'. Each example is described by k binary features $f(x, y)$. In the learning phase, each feature is assigned with its weight α.

The notion of a feature is different from the common understanding of the term in other machine learning methods. Here, a feature is a function $f : \mathcal{A} \times \mathcal{B} \to \{0, 1\}$, where $\mathcal{A}$ denotes the set of output classes and $\mathcal{B}$ denotes the set of possible contexts. After introducing the contextual predicate $cp : \mathcal{B} \to \{true, false\}$, which is a function indicating the presence of some information in the given context $b \in \mathcal{B}$; Ratnaparkhi [1998] defines a feature as

$$f_{cp,a'}(a, b) = \begin{cases} 1 & \text{if } a = a' \text{ and } cp(b) = true \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

Thus all the features of the model are binary, and all of them are functions of both the context and the output variable.

In order to estimate the parameters of the model ($\{\alpha_i\}_1^k$), Ratnaparkhi [1998] uses an iterative algorithm known as general iterative scaling (GIS).

There are two main differences between (binary) logistic regression and the maximum entropy framework: firstly, the number of classes being predicted and secondly, the type of features used for describing the examples (because logistic regression uses real-valued explanatory variables irrespective of the output class). Ratnaparkhi [1998, pp. 27–28] shows that if one enables real-valued features and limits the number of predicted classes, the maximum entropy model (2.9) is equivalent to logistic regression (2.1). Assuming the new type of features is defined as

$$\begin{aligned} f_0(a, b) &= \begin{cases} 1 & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases} \\ f_j(a, b) &= \begin{cases} x_j & \text{if } a = 1 \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{2.11}$$

where x_j is a value of the j^{th} explanatory variable; the logistic regression model

(2.1) can be derived as follows:

$$
\begin{aligned}
p(y=1|x) &= \frac{\prod_{j=0}^{k} \alpha_j^{f_j(1,x)}}{\prod_{j=0}^{k} \alpha_j^{f_j(0,x)} + \prod_{j=0}^{k} \alpha_j^{f_j(1,x)}} \\
&= \frac{\prod_{j=0}^{k} \alpha_j^{f_j(1,x)}}{1 + \prod_{j=0}^{k} \alpha_j^{f_j(1,x)}} \\
&= \frac{e^{\theta^\mathsf{T} x}}{1 + e^{\theta^\mathsf{T} x}} = \frac{1}{1 + e^{-\theta^\mathsf{T} x}}.
\end{aligned} \tag{2.12}
$$

The third equality results from setting $\theta := (\ln(\alpha_0), \ln(\alpha_1), \ldots, \ln(\alpha_k))$ and from the fact that

$$
\prod_{j=0}^{k} \alpha_j^{f_j(1,x)} = \exp(\sum_{j=0}^{k} \ln(\alpha_j) f_j(y,x)) = \exp(\sum_{j=0}^{k} \ln(\alpha_j) x_j).
$$

2.1.3 Logistic regression and multi-class classification

Ratnaparkhi [1998, p. 28] states that "logistic regression models are designed for problems with binary-valued outcomes, and are not suited for natural language tasks [...]". While this might be true for tasks such as part of speech tagging, where the target variable has many levels, it is less a problem in the case of three output classes since there are ways to combine different binary classifiers to produce multi-class classification. Moreover, there is a generalization of logistic regression handling the multi-class classification directly.

Multinomial logistic regression

Multinomial logistic regression is an extension of the binary case. The sigmoid function used in the binary model (2.1) is replaced by its generalization — softmax function — to handle multiple classes:

$$
h_\theta(x) = p(y=c|x) = \frac{\exp(\theta^{(c)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)}. \tag{2.13}
$$

K denotes the number of classes, c an arbitrary class and $\theta^{(c)}$ a set of parameters associated with the class c. Unlike in the case of binary logistic regression, here

the set of model parameters forms a matrix:

$$\theta = \begin{bmatrix} \theta_0^{(1)} & \theta_0^{(2)} & \dots & \theta_0^{(K)} \\ \theta_1^{(1)} & \theta_1^{(2)} & \dots & \theta_1^{(K)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_n^{(1)} & \theta_n^{(2)} & \dots & \theta_n^{(K)} \end{bmatrix} \tag{2.14}$$

Each row of the matrix corresponds to a single feature and each column corresponds to a specific output class.

In order to adapt the training phase to the new model, the cost function from the binary model (2.3) is replaced by

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} I(y_i = k) \log \frac{\exp(\theta^{(k)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} + \frac{1}{m} \lambda \Omega. \tag{2.15}$$

I is the indicator function signaling whether its argument is true or not. By using the indicator function together with the summation over all K classes, the cost function considers for each example only the probability the model assigns to the correct class y. The regularization term Ω also needs to be changed to include all the weights of the model:

$$\Omega = \sum_{i=1}^{n} \sum_{j=1}^{K} \theta_{ij}^2. \tag{2.16}$$

Again, the form of the cost function is justified by its relationship to the likelihood, which for this model is expressed as

$$\mathcal{L}(\theta|x) = P(x|\theta) = \prod_{i=1}^{m} \frac{\exp(\theta^{(y_i)\top} x_i)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x_i)}. \tag{2.17}$$

Now the cost can be expressed as $-\frac{1}{m} \log \mathcal{L}(\theta|x) + \frac{1}{m} \lambda \Omega$. To estimate the parameters, the same algorithm (LBFGS) is used as in the case of binary logistic regression.

When the term "logistic regression" is used more freely to refer also to the multinomial case described here, both logistic regression and maximum entropy model are treated as equivalent by some authors [Jurafsky and Martin, 2009]. With respect to the description by Ratnaparkhi [1998], the single difference is the nature of the features. Under the assumptions in (2.11), the form of multinomial regression can be derived from the maximum entropy model along the lines shown by (2.12).

One vs. all approach

The problem of multinomial classification can be approached from another angle. Instead of changing the form of the model, multiple binary classifiers can be

trained on different subsets of the general problem and then combined together to produce the final outcome. This type of approach is not limited to logistic regression and can be used with any set of binary classifiers. (In fact, it is often used with support vector machines that are "inherently two-class classifiers." [Manning et al., 2008, p. 330])

In the one-vs-rest approach, a single binary classifier is trained for each class to discriminate between that class on the one side and the other classes on the other side. At the time of classification, all the models are run on the given example and the label that achieves the highest score is selected as the final outcome.

Similarly to the multinomial logistic regression, this algorithm learns K sets of parameters (where K is the number of classes). Here however, the parameters are estimated separately on modified data, which can lead to different results as shown in Figure 2.1. A disadvantage of this approach is that by dividing the training examples into a single class and its complement, one can create highly imbalanced data.

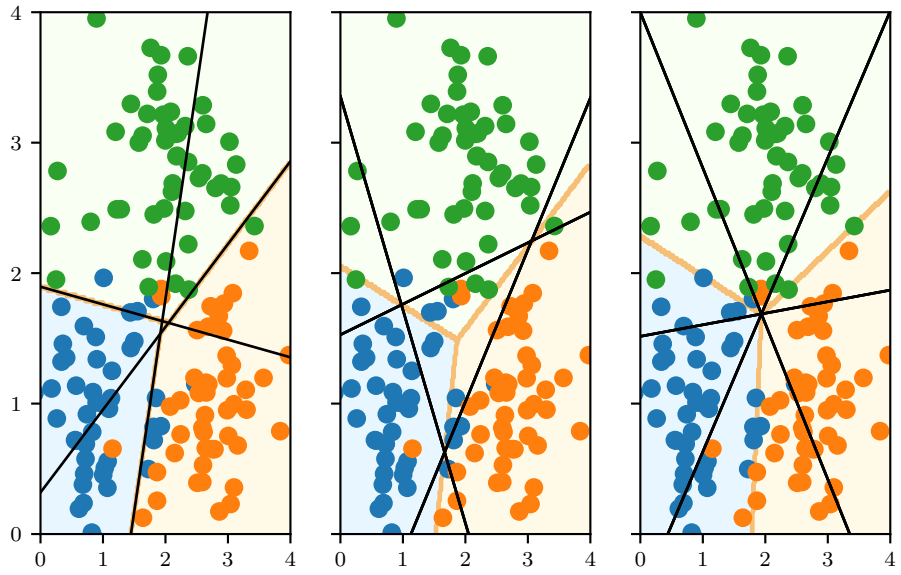


Figure 2.1: Comparison of decision boundaries of the one-vs-one, one-vs-rest and multinomial approaches (from left to right). Black lines mark the boundaries of individual classifiers while the colored surfaces represent the prediction of the overall models. Data was created by taking 50 random samples from each of the three bivariate Gaussian distributions: $\mathcal{N}_2(\begin{bmatrix}1\\1\end{bmatrix}, \Sigma)$, $\mathcal{N}_2(\begin{bmatrix}3\\1\end{bmatrix}, \Sigma)$, and $\mathcal{N}_2(\begin{bmatrix}2\\3\end{bmatrix}, \Sigma)$, where $\Sigma = \begin{bmatrix}0.3 & 0\\0 & 0.3\end{bmatrix}$. The accuracy scores of the fitted models are 0.907 for one-vs-one, 0.894 for one-vs-rest and 0.927 for multinomial approach.

One vs. one approach

Another approach of combining binary classifiers to solve multinomial classification problem is to train a single classifier for each pair of classes. In total one needs to fit $\binom{K}{2}$ models, which can be costly if the number of classes is large. However, in the case of three classes, the number of models is the same as for the one-vs-rest approach.

The final outcome of the set of binary models can be in this case determined by simple major vote, i.e. selecting the class that was predicted most often. The obvious disadvantage of this approach is that sometimes ties may occur. In the case of three-class problem, this happens when each model predicts a different class. To mitigate this problem, Hastie and Tibshirani [1998] suggested an algorithm known as pairwise coupling.

Unlike in the case of the one-vs-rest approach, the problem with a set of pairwise classifiers is that the output they produce cannot be compared directly as each output represents $P(A_i|A_i or A_j)$, where A_i and A_j belong to a set of K classes $A_1, A_2, \dots A_K$. Setting $K := 3$, the individual outputs for one observation can be represented by a 3 by 3 matrix:

$$\begin{bmatrix} . & r_{1,2} & r_{1,3} \\ r_{2,1} & . & r_{2,3} \\ r_{3,1} & r_{3,2} & . \end{bmatrix}$$

where each $r_{ij} = \frac{p_i}{p_i + p_j}$. In the matrix, each pair (r_{ij}, r_{ji}) sums to 1. The task is to estimate the probabilities $p_i = P(A_i)$. Setting n_{ij} to be the number of observations based on which the probability r_{ij} was estimated, i. e. the number of examples in the training data where the target class is either A_i or A_j, Hastie and Tibshirani [1998] propose estimating the probabilities by finding parameters of the binomial model

$$\begin{aligned} n_{ij} r_{ij} &\sim \text{Binomial}(n_{ij} \mu_{ij}) \\ \mu_{ij} &= \frac{p_i}{p_i + p_j}. \end{aligned} \tag{2.18}$$

To find the probabilities p_i, they suggest an iterative algorithm that makes the initial guess of $\hat{p}_i$ and μ_{ij} at each step closer to r_{ij}. This is done by minimizing the Kullback-Leibler distance between the theoretical and empirical distribution:

$$\begin{aligned} \ell(p) &= \sum_{i \neq j} n_{ij} r_{ij} log \frac{r_{ij}}{\mu ij} \\ &= \sum_{i<j} n_{ij} \left[r_{ij} log \frac{r_{ij}}{\mu ij} + (1 - r_{ij}) log \frac{1 - r_{ij}}{1 - \mu ij} \right] \end{aligned} \tag{2.19}$$

By setting its derivative

$$\frac{\partial \ell(p)}{\partial p_i} = \sum_{j \neq i} \left(-\frac{r_{ij}}{p_i} + \frac{1}{p_i + p_j} \right) \tag{2.20}$$

to zero and multiplying by p_i, the following equation derived:

$$\sum_{j \neq i} n_{ij} \mu_{ij} = \sum_{j \neq i} n_{ij} r_{ij} \tag{2.21}$$

To find the point that satisfies (2.21) and $\sum_{i=1}^{K} p_i = 1$, Hastie and Tibshirani [1998] suggest the following algorithm:

1. Start with an initial guess on $\hat{p}_i > 0, \forall i$ and corresponding $\hat{\mu}_{ij} = \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j}$.
2. Repeat until convergence for each class consecutively ($i = 0, 1, \ldots K, 1, \ldots$):
   (a) compute the new value of $\hat{p}_i := \hat{p}_i \frac{\sum_{j \neq i} n_{ij} r_{ij}}{\sum_{j \neq i} n_{ij} \hat{\mu}_{ij}}$,
   (b) normalize the new value $p_i := \frac{\hat{p}_i}{\sum_{j=1}^{K} \hat{p}_j}$,
   (c) based on the new p_i, compute the new estimates of μ_{ij}: $\hat{\mu}_{ij} = \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j}$.

Hastie and Tibshirani [1998] prove that the above procedure converges, which results in stable estimates of $P(A_i)$ for each class $A_1, A_2, \ldots A_K$. Based on these estimates, one can predict a new example by selecting the class with the highest probability $\hat{p}$.

Besides the more involved process of combining the prediction of the individual binary models, another disadvantage of the one-vs-one approach is the fact that each classifier is used for also predicting examples it did not see during training (e.g. for an observation with true output A_k, a model trained to distinguish between A_i and A_j also needs to be used). A related problem is a lower number of training examples for each binary classifier when compared to one-vs-rest approach. On the other hand, the models in one-vs-one approach do not suffer from artificially introduced imbalance to the data. The way one-vs-one method can differ from the previous ones in terms of the extracted decision boundaries can be seen in Figure 2.1.

While sometimes the one-vs-one and one-vs-all approaches are considered "not very elegant" [Manning et al., 2008, p. 330], in practice, they can work well as will be shown by the experiments in a later section of the thesis.

2.2 Gradient tree boosting & XGBoost

As an alternative to logistic regression, extreme gradient boosting [Chen and Guestrin, 2016], better known as "XGBoost", was used. XGBoost is a specific

implementation of gradient tree boosting, an algorithm introduced by Friedman [2000]. The general idea of boosting is to build an ensemble of models, where each model performs slightly better than chance (the models are often called 'weak learners'). However, each model in the sequence is trained in such a way, that it learns to improve upon the prediction of the previous ones. Together, the collection of such models produces a composite model with (hopefully) accurate predictions (the so called 'strong learner').

XGBoost has been shown to achieve state-of-the-art results in many machine learning problems, as illustrated by top scoring solutions in competitions such as Kaggle. Moreover, the fact the implementation is scalable by enabling parallel tree learning and by effective handling of sparse data makes it also a good choice for natural language processing applications, determiner prediction included.

In general terms, the XGBoost model is defined as a collection of functions whose outputs are summed to produce the final prediction:

$$h(x) = \sum_{k=1}^{K} f_k(x), f_k \in \mathcal{F}. \tag{2.22}$$

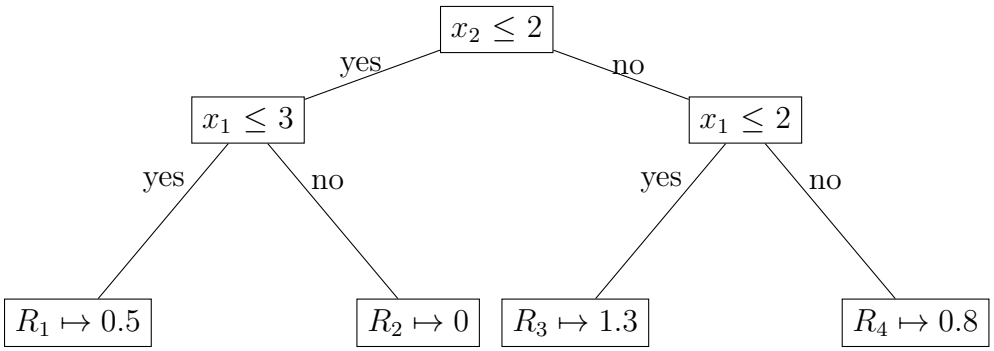
Here, x stands for the feature vector corresponding to the example being predicted; $h(x)$ is the model hypothesis or prediction for the particular example; f is a function from a predefined functional space $\mathcal{F}$. While XGBoost implementation is general enough to provide more options for $\mathcal{F}$, the standard choice, taken also in this thesis, is to define $\mathcal{F}$ as a set of possible regression trees.

2.2.1 Regression trees

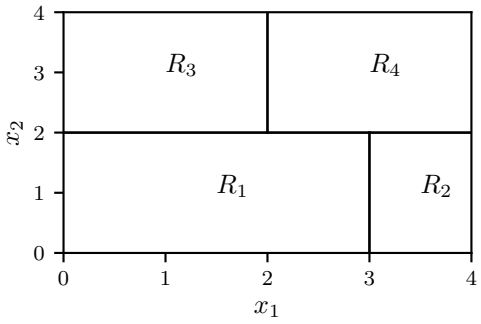
A regression tree[4] is a machine learning model on its own; it is a function that produces a real-valued outcome for each feature vector representing an observation. The function is represented as a binary tree, where each node is associated with a splitting criterion on the data: a specific feature from the feature set and a specific value (Figure 2.2a). By comparing data examples to the selected value of the selected feature, the space of all possible observations is partitioned into two distinct regions corresponding to the two children of the node. An observation is processed by following a path from the tree root to one of its leaves based on the conditions stored in each inner node. Each leaf of the tree represents a region of the feature space, formed by all the data points that would follow the same path in the tree (Figure 2.2b). Lastly, each leaf is assigned with a real-valued constant

[4]There are more variants of regression trees. The ones used by XGBoost and described here are the so called 'CART' trees (where 'CART' stands for 'classification and regression tree') proposed by Breiman et al. [1984].

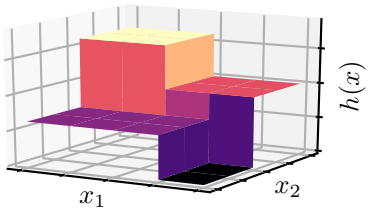
— the output of the function — shared by all the observations belonging to the corresponding region (Figure 2.2c).



(a) regression tree



(b) regions



(c) output values

Figure 2.2: An example of a regression tree for two–dimensional input data $\boldsymbol{x} = (x_1, x_2)$. 2.2a shows the graph representation of the tree; 2.2b illustrates the final partitioning of the feature space; prediction values for each region are shown in 2.2c. (The illustration is adapted from a similar example by [Hastie et al., 2009, p. 306])

Let c_i denote the constant corresponding to the region (leaf) R_i out of the total number of J such regions (leaves). Then the prediction of the regression tree can be formally written as

$$h(x) = \sum_{i=1}^{J} c_i I(x \in R_i), \tag{2.23}$$

where I is the indicator function signaling whether its argument is true or not.

Training

The parameters of the model described in (2.23) are the structure of the tree (i.e. the nodes with the splitting criteria), which is represented by I in the equation; and the set of output values $\{c_i\}_1^J$.

These parameters are estimated from the training data by a greedy algorithm. The tree is built by a top–down approach, i.e. starting at the root and finishing at the leaves. For each node, a splitting criterion is selected out of all possible features and their values. The objective is to split the data into two sets that are, with respect to the target variable, as homogeneous as possible. The quality of the split is determined by the mean squared error:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - h'(x_i))^2, \tag{2.24}$$

where N is the number of the training observations and h' is the current model (i.e. the tree built so far, including the split being evaluated). The output values $\{c_i\}_1^J$ corresponding to the regions $\{R_i\}_1^J$ are computed as the mean of the output variable of all the observations contained within the specific region:

$$c_i = \frac{1}{|R_i|}\sum_{j=1}^{|R_i|}(y_j | x_j \in R_i). \tag{2.25}$$

The same procedure of finding the best data split is applied recursively to the resulting sets from the previous steps (Figure 2.3) until a stopping criterion is met. This might be based on the number of observations with respect to the leaves, or on a threshold on a node's mean squared error beyond which the node is not split any further.

2.2.2 Gradient tree boosting

After a brief detour to regression trees, the following sections continue with the description of a specific form of their ensembles: gradient boosting and the XGBoost algorithm. The presented form of the model (2.22) i.e. a sum of base (weak) learners is not specific to boosting only. What distinguishes boosted trees from models such as random forests is the way of building the ensemble. Friedman [2000] presents gradient boosting as an analogy to numerical optimization methods, where the estimated set of parameters P^* can be expressed as a sum of sequential increments $\{p_m\}_0^M$ (p_0 being an initial guess):

$$P^* = \sum_{m=0}^{M} p_m. \tag{2.26}$$

Similarly, the process of building a boosted tree ensemble can be seen as "numerical optimization in function space" [Friedman, 2000, p. 3], expressed as a sequence of incremental boosts to the previous state of the model:

$$F^* = \sum_{m=0}^{M} f_m(x). \tag{2.27}$$

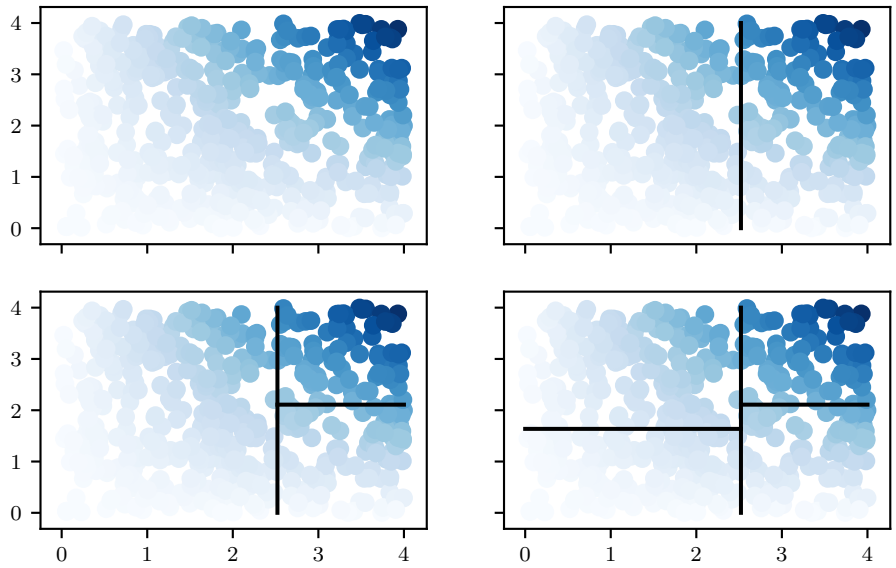


Figure 2.3: First steps of the greedy algorithm minimizing the mean squared error of a regression tree on an artificially generated dataset (500 observations of two uniform random variables: $X_1 \sim \mathcal{U}(0,4), X_2 \sim \mathcal{U}(0,4), Y = X_1 X_2$). From top left to bottom right the MSE values are 12.1, 7.8, 4.5 and 3.2.

However, unlike in the case of the standard estimation of parameters by numerical optimization, here the parameters (functions) are estimated in a stagewise fashion, i.e. one parameter at a time without modifying the parameters that have already been estimated. (When searching for the parameters of a logistic regression model or a neural network, the algorithms take also many steps, but at each step, all the parameters are fit jointly).

Concretely, the prediction of the target variable $\hat{y}_i$ for the i^{th} observation x_i at the step t can be expressed with respect to the prediction in the previous step as:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i). \tag{2.28}$$

At each step t, one tries to find a tree f_t that makes the predictions of the new ensemble as close to the observed values as possible. This is achieved by minimizing a cost function, which can be, depending on the task, the mean squared error (2.24), logistic loss (2.3), multinomial logistic loss (2.15) and many others.

2.2.3 XGBoost

XGBoost extends on the original tree boosting algorithm by employing additional regularization term. Thus the actual regularized cost function at the step t is

defined as:

$$\begin{aligned}\mathcal{L}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i) \end{aligned} \tag{2.29}$$

where n is the number of training examples, l is an arbitrary differentiable cost function (such as mean squared error) and Ω is a measure of tree complexity. Informally, minimizing such a function should result in a model that fits the training examples well without modeling too much of the random noise in the data so that it can generalize on unseen examples.

Furthermore, for each training example (x_i, y_i), the cost function l is approximated by its Taylor series[5] up to the second order at the point of the previous prediction $\hat{y}_i^{(t-1)}$:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \sum_{i=1}^{t} \Omega(f_i) \tag{2.30}$$

Here, g_i and h_i are defined as:

$$\begin{aligned} g_i &= \frac{\partial l}{\partial \hat{y}_i^{(t-1)}}(y_i, \hat{y}_i^{(t-1)}) \\ h_i &= \frac{\partial^2 l}{\partial \hat{y}_i^{(t-1)}}(y_i, \hat{y}_i^{(t-1)}) \end{aligned}$$

This approximation enables the algorithm to handle any user-defined loss function l while staying computationally efficient. Moreover, since both $l(y_i, \hat{y}_i^{(t-1)})$ and $\sum_{i=1}^{t-1} \Omega(f_i)$ are constant at the step t, it is enough to minimize the simplified form of the regularized cost function:

$$\widetilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_i). \tag{2.31}$$

The regularization term is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2, \tag{2.32}$$

where T is the number of tree leaves, w_j is the predicted value for the j^{th} leaf of the tree and γ and λ are coefficients regulating the strength of the regularization term with respect to the tree size and the prediction values. (Both last coefficients serve as hyperparameters of the model). Unfortunately, the choice of the specific form of the regularization term (2.32) is not justified by Chen and Guestrin [2016].

[5] See Section 2.1.1 for its definition.

By expanding the regularization term (2.32) and setting $I_j = \{i|x_i \in R_i\}$, i.e. set of all example indices belonging to the region (leaf) j, the cost function can be further rewritten as:

$$\begin{aligned}\widetilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \end{aligned} \tag{2.33}$$

For a given tree, the best estimate for the output value[6] w_j can be found analytically by taking the derivative of the relevant part of the loss function and setting it to zero: $((\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2)' = 0$. Solving the equation for w_j produces the following result

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}. \tag{2.34}$$

The formula also illustrates the impact the regularization coefficient λ has on lowering the predicted values of the tree.

After substituting the leaf values in (2.33) with their estimates from (2.34) and simplifying, the ultimate regularized loss function of the whole tree is expressed as

$$\widetilde{\mathcal{L}}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{2.35}$$

The derived cost function can be used to evaluate an existing tree. It is impossible to enumerate all possible trees and compare their cost. Instead, as mentioned in the section on the training of regression trees (2.2.1), a greedy top-down algorithm is used. In order to compare potential splits at each stage of the process of building the tree, the cost function (2.35) is utilized to create a measure of a split gain:

$$SplitGain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma. \tag{2.36}$$

Here, I stands for all the examples belonging to the potential split node, I_L and I_R represent the examples belonging to its left and right child (given the split is chosen). Thus the cost *reduction* of a split can be interpreted as the difference between the cost of the node before the split and the cost of the two new nodes after the split: $(-\frac{1}{2}\frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i+\lambda}+\gamma) - (-\frac{1}{2}\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda}+\gamma-\frac{1}{2}\frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i+\lambda}+\gamma)$. Additionally, this form of the split gain equation shows explicitly the role of the

[6] Chen and Guestrin [2016] use the term 'weight' for the output of a leaf

regularization parameter γ. It prevents the algorithm from selecting splits that do not result in cost reduction that is 'large enough', namely those whose reduction is less than γ.

The provided derivation of the split finding criterion illustrates the differences between XGBoost and the original algorithm of gradient boosted trees. Using the second order approximation of the original loss function l (2.30) makes it possible to have a single implementation for any user-defined loss function (as long as l' and l'' are provided). Moreover, by adding the tree complexity penalty to the cost function (2.32), regularization is embedded to the core of the tree construction.

3. Experimental Setup

The conducted experiments were designed to firstly replicate and secondly to improve upon the experiment by Lee [2004] who, in turn, followed rather closely the experiment by Minnen et al. [2000]. The reason for such strong adherence to other research (i.e. the replication step) is the desire to produce truly comparable results. Lee [2004] provides rather clear description of experiment settings and, on the given corpus, reaches the best results known to the author. In this chapter, sources of data are described as well as the features used by the classifiers.

3.1 Data

3.1.1 Penn Treebank

Because of the above-mentioned replication step, the basis for the classifiers trained for the thesis is the Penn Treebank corpus.[1] It contains a collection of manually parsed newspaper articles from the business-focused Wall Street Journal. There is about one million words in the corpus. The articles are divided into 24 sections. The treebank has been especially popular with parsing research which settled on de facto standard division of the data into the training, held-out and parsing parts. This division was accepted by Lee [2004] and also by this project. Sections 00 – 21 constitute the training, 22 the held-out, and 23 the testing part.

Although the treebank data are manually tagged and parsed, Lee [2004] chose to modify the training data by performing automatic preprocessing from the raw text. For part of speech tagging, the tagger by Ratnaparkhi [1998] was used. Then the data was parsed by the Collins parser [Collins, 2010]. Lee [2004] advocates the use of the Penn Treebank data as a means for facilitating the comparison with previous research, namely that by Knight and Chander [1994] and Minnen et al. [2000]. However, it is not clear why the experiment differs from those articles by not using manual parses. Since Lee [2004] outperforms the other two articles, the same approach is adopted here in order to ensure that potential differences in the results are not caused by the fact better training data are used.[2]

From the training data, all the base noun phrases are extracted. A base noun

[1] The Penn Treebank, version 3. 1999. Distributed by Linguistic Data Consorcium. URL: `https://catalog.ldc.upenn.edu/ldc99t42`

[2] In fact, the Collins parser was trained on the Penn Treebank corpus, thus when it is used to parse its own training data, the parses are likely to be more accurate than if different data was used.

phrase can be defined as a noun phrase that does not dominate any other non possessive noun phrase. [Vadas and Curran, 2007] Examples of base noun phrases are labeled by the 'NPB' tag in the following phrases:

a) *NP(NPB(NN(chairman)) PP(IN(of) NPB(NNP(Elsevier) NNP(N.V.))))*

b) *NPB(NPB(NNP(Boston) POS('s)) NNP(Dana-Farber) NNP(Cancer) NNP(Institute))*

While base noun phrases are marked as such by the Collins' parser in the training data, they have to be specifically extracted from the test data.[3] There is no non-base noun phrase that would contain an article as its direct child in the test data.

Apart from identifying noun phrases, head of phrases had to be extracted because — as will be described later — it is an important piece of information for many features. Again, Collins parser identifies the heads of phrases automatically. For the test data, the heads were extracted using the head finding rules given by Collins [2010].

Much syntactic ambiguity stems from the fact that the hierarchy of a sentence is manifested only as a linear sequence of words.[4] In the context of determiner prediction, this can also play a role, for example in phrases containing a possessive noun phrase. Considering the string *"a Bigg's hypermarket"*, its syntactic interpretation can be either of the following:

a) *NP(DT(a) NP(NNS(Bigg) POS('s)) NN(hypermarket))*

b) *NP(NP(DT(a) NNS(Bigg) POS('s)) NN(hypermarket))*

From the utilitarian perspective of a grammar checker, it is not important to which phrase the article is assigned. However, this fact is not reflected in evaluation since the task is defined as classification of noun phrases. Moreover, there is only one possessive noun phrase with an article belonging to the parent NP in the test data (namely the example a) above). The Collins' parser does not seem to consider the distinction either since there is not a single such example in the automatically parsed training data.

[3] Unfortunately, this step of collecting the test data is not mentioned by Lee [2004]. Nevertheless we assume it to be the case and proceed with such a test set. This decision is also supported by the fact that models evaluated on all noun phrases (not just on base noun phrases) achieve higher scores. Thus, it is granted that an improvement would not be achieved by using different evaluation data.

[4] This is true for written texts. In speech, there might be other indicators of sentence structure, such as intonation and other prosodic phenomena.

From the training set, about 263 000 noun phrases were extracted. The held-out and test sets are formed by about 10 000 and 14 000 extracted phrases respectively. The distribution of the articles over the data is shown in Table 3.1. It also shows the baseline accuracy of a model that always predicts zero article – 0.711.

	train	held-out	test
zero	70.7%	69.7%	71.1%
the	19.8%	21.5%	19.7%
a/an	9.5%	8.8%	9.2%

Table 3.1: Distribution of the articles (determiners) over the data sets.

3.1.2 British National Corpus

For learning the countability information of nouns based on their context the British National Corpus was used.[5] The portion of the corpus consisting of written texts was parsed by the Stanford PCFG parser [Klein and Manning, 2003]. The data were shuffled and divided into the training, held-out and test set in the ratio 8 : 1 : 1. There are about 6.9 million tokens in the training set.

3.1.3 One Billion Word Benchmark

For learning a language model, One Billion Word Benchmark was used [Chelba et al., 2013]. This is a large corpus of tokenized English sentences (in randomized order) intended specifically for comparisons of different approaches to language modeling. The sentences were selected from the monolingual English dataset originally provided by the sixth workshop on statistical machine translation [Callison-Burch et al., 2011]

The corpus contains about 0.8 billion word tokens and there is about 0.8 million words in the vocabulary. The same division of the data to train, held-out and test portions is used as in [Chelba et al., 2013].

3.2 Features

In this section, the features used for the classifier are described. Conceptually, the features are divided into several groups that are then used for evaluation. The

[5] The British National Corpus, version 3 (BNC XML Edition). 2007. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: `http://www.natcorp.ox.ac.uk/`

later groups serve as an extension not as a substitution of the previous ones. In what follows, each group, its label for further reference, and a list of corresponding features are given.

3.2.1 Original features

label: ORIG

As mentioned earlier, the experimental part of the thesis starts with an attempt to replicate the experiment by Lee [2004]. The set of the features used in that work is given here. Some of the attributes were introduced in older research [Minnen et al., 2000], others are the contribution of the experiment being replicated.

- **head form**

  lemma of the head of the noun phrase;

  e.g. hypermarket — a Bigg's hypermarket

- **head part of speech**

  part of speech of the head of the noun phrase, without the number information (see below);

  e.g. NN — a Bigg's hypermarket

- **head number**

  grammatical number of the noun phrase head;

  e.g. singular — a Bigg's hypermarket

- **parent**

  part of speech of the parent of the noun phrase;

  e.g. PP — The average seven-day simple yield of NP(the 400 funds) was 8.12%.

- **words before head**

  set of lemmas of words that precede the head and belong to the noun phrase, excluding articles;

  e.g. [average, seven-day, simple] — NP(The average seven-day simple yield) of the 400 funds was 8.12%.

- **words after head**

set of lemmas of words that follow the head and belong to the noun phrase, excluding articles;

e.g. None — NP(The average seven-day simple yield) of the 400 funds was 8.12%.

- **parts of speech before head**

  set of parts of speech of words that precede the head and belong to the noun phrase, excluding articles;

  e.g. [JJ, JJ, JJ] — NP(The average seven-day simple yield) of the 400 funds was 8.12%.

- **parts of speech after head**

  set of parts of speech of words that follow the head and belong to the noun phrase, excluding articles;

  e.g. None — NP(The average seven-day simple yield) of the 400 funds was 8.12%.

- **words before NP**

  set of lemmas of two words before the noun phrase, excluding articles;

  e.g. [yield, of] — The average seven-day simple yield of NP(the 400 funds) was 8.12%.

- **words after NP**

  set of lemmas of two words after the noun phrase, excluding articles;

  e.g. [be, <number>] — The average seven-day simple yield of NP(the 400 funds) was 8.12%.

- **non-article determiner**

  other determiner (not including articles);

  e.g. no — There's NP(no question) about ...

- **hypernyms**

  the hypernyms for the first synset (sense) of the head, as extracted from the WordNet [Miller, 1995];

  e.g. questioning — There's NP(no question) about ...

- **referent**

  an indicator whether the given head appeared in any of the five previous sentences;

3.2.2 Extended original features

label: Ext

The features mentioned in this section are either a simple modification of the previous features or they can be understood as directly inspired by them. There is no additional source of information used for extracting the features, just the text itself and syntactic parses.

- **head part of speech - simple**

  part of speech tag of the head where the distinction between proper and common nouns is not present (in addition to the original simplification where grammatical number is not taken into account);

  this feature replaces the head part of speech feature from the previous set, i.e. they are not used together for any trained model;

  e.g. NN — chairman of NP(the Prudential Insurance Co.)

- **head proper**

  the information discarded in the previous feature: an indicator whether the head of the noun phrase is a proper noun;

  e.g. true — chairman of NP(the Prudential Insurance Co.)

- **words before head**

- **words after head**

- **parts of speech before head**

- **parts of speech after head**

- **words before NP**

- **words after NP**

  The six features above copy the identically titled features from the original set with the exception that the word sequences are not taken as sets of separate words but as fixed n-grams.

  e.g. (words before head feature) average seven-day simple — the average seven-day simple yield

- **referent**

  an extension of the same feature from the previous set, an indicator whether the given head appeared in any of the five previous sentences or in the current sentence up to the occurrence of the head;

this feature replaces the referent feature from the previous set, i.e. they are not used together for any trained model;

- **postmodification type**

  part of speech of the following sibling of the noun phrase in case the noun phrase is a child of another noun phrase;

  e.g. PP — NP(\$500 million) of Remic mortgage securities

- **object - main verb form**

  in case the noun phrase is identified as an object of a verb (its parent is a verb phrase) the feature corresponds to the lemma of the verb form;

  e.g. cause — ... will undoubtedly cause NP(renewed debate)

- **object - preposition**

  in case the noun phrase is identified as an object of a prepositional phrase (its parent is a prepositional phrase), the feature corresponds to the preposition;

  e.g. of — At the end of NP(the day)

3.2.3 Countability feature

label: COUNT

The information about the grammatical category of countability of a given noun phrase head is not directly available from the text data. However, as the mass — count distinction determines what articles can be used with the given noun it seems as an important feature for the classification problem at hand.

In the previous research, some authors did not attempt to extract such a feature [Turner and Charniak, 2007], [Lee, 2004]; others extract it simply from a dictionary [Dahlmeier et al., 2013]; still others attempt to label nouns as mass or count by comparing the ratio of singular and plural forms of the noun within a corpus [Han et al., 2006]. For this thesis a more involved approach by Nagata et al. [2005] is adopted. They report achieving 93% accuracy on distinguishing nouns as mass or count in the writings of Japanese learners of English. The authors understand the problem as word sense disambiguation since a noun can be either count or mass depending on the sense the noun is used in. Therefore, they built a classifier that labels the nouns based on other words appearing in the immediate context.

Decision lists classifier

At first, some head nouns are extracted from a parsed corpus, namely those that can be identified as count or mass by applying a small set of rules (e.g. *Is the noun in plural? (count noun); Does it appear with an indefinite article? (count noun); Is the noun modified by "much"? (mass noun);* for the complete list of rules, see [Nagata et al., 2005, p. 817]). The nouns are extracted together with three types of contexts: all the words within the noun phrase, three words preceding the noun phrase and three words following the noun phrase (articles, and some other function words are excluded, context words are lemmatized and lowercased). Each word from each context then forms a rule together with the corresponding context type, head noun and the extracted label (i.e. count/mass). For example, for the head noun *chicken* in the sentence *she ate a piece of fired chicken for dinner*, three decision rules would be extracted: $\text{piece}_{-3} \rightarrow$ Mass; $\text{fry}_{np} \rightarrow$ Mass; $\text{dinner}_{+3} \rightarrow$ Mass; (the subscripts identify the context type). [Nagata et al., 2005, p. 819] All the rules are then sorted into a decision list by their log likelihood ratio defined by:

$$log\frac{p(MC|w_c)}{p(\overline{MC}|w_c)}, \tag{3.1}$$

where MC is the variable denoting the mass or count label, and $p(MC|w_c)$ is the probability of the head noun occurring with the MC label with the word w in the context c. The probability is estimated from the corpus by:

$$p(MC|w_c) = \frac{f(MC, w_c) + \alpha}{f(w_c) + 2\alpha}, \tag{3.2}$$

where $f(MC, w_c)$ is the frequency of the head noun labeled as MC while having word w in the context c; α is a smoothing parameter and is set to 0.5. In addition, each decision list is extended by one rule that assigns the head noun the most frequent label MC_{major} regardless of any context. The score of this default rule is given by

$$log\frac{p(MC_{major})}{p(\overline{MC_{major}})} \tag{3.3}$$

where $p(MC_{major})$ is estimated by:

$$p(MC_{major}) = \frac{f(MC_{major})) + \alpha}{f(MC_{major}) + f(\overline{MC_{major}}) + 2\alpha} \tag{3.4}$$

where $f(MC_{major})$ and $f(\overline{MC_{major}})$ correspond to the frequency of the target noun appearing with the more and less frequent label respectively, the smoothing parameter α is set to 0.5.

An illustration of a possible decision list for the target noun *chicken* is given by table 3.2.

rule			
context word	context type	label	log-likelihood ratio
piece	-3	Mass	1.49
count	-3	Count	1.49
peck	+3	Count	1.32
fish	-3	Mass	1.28
dish	-3	Mass	1.23
pig	np	Count	1.23

Table 3.2: Decision list for determining countability of the noun *chicken*; taken from Nagata et al. [2005]

When a new noun phrase is encountered, the rules from the corresponding decision list are checked from top to bottom. The first rule that can be applied to the target noun and its context determines the final label of the noun. In case more rules share the same score, the label predicted by more decision rules is used. In case both labels are predicted by the same number of rules with the same score, following rules in the list are taken into account. If the default rule (assigning the target noun with a label regardless of context) is part of the tie, its label is used. The default rule is the only rule that can be applied in any context, therefore rules with lower scores are discarded and the decision must be made when the default rule is met in the list.

Data

As mentioned in the section on used data sources, decision lists are trained on the training section of the British National Corpus. Detailed information is given in Section 3.1.2. There were about 260 000 different lemmas for which its countability could be guessed at least once based on the rules in [Nagata et al., 2005]. For each of those lemmas a decision list was created.

3.2.4 Word embeddings

label: EMB

Another source of information for the classifier that cannot be extracted from the training corpus on its own is the word embedding representation of the head of the noun phrase.

Motivation

A word embedding is a representation of a word that maps the word into a multidimensional space. The mapping is trained in such a way that words occurring within similar contexts are closer to each other in the vector space than words occurring in different contexts. Moreover, the well-known property of this representation is that some linguistic patterns can be expressed by simple linear operations on these vectors, e.g. "vec('Madrid') - vec('Spain') + vec('France') is closer to vec('Paris') than to any other word vector" [Mikolov et al., 2013b, p. 1]

Even though no one-to-one relationship between an embedding dimension and a certain linguistic phenomenon has ever been described; we try to use the individual dimensions of the embeddings as features in the classifier, hoping the classifier could benefit at least from some of the features.

Word2vec

There are more ways the word vectors can be obtained. For this thesis, we select the architecture known as word2vec, proposed by Mikolov et al. [2013b]. It is based on the Skip-gram model introduced earlier by Mikolov et al. [2013a] that is made more efficient so that it enables learning the word vectors on large data. The authors train a neural network with one hidden layer to predict the probabilities of different words co-occurring with the input word within a context window. The architecture of the neural network is shown in figure 3.1.

Each node in the input layer corresponds to one word in the vocabulary. The words are represented in one-hot encoding, i.e. by vectors whose length is the same as the size of the vocabulary. In each vector, there is exactly one value (corresponding to the word) set to 1, all other values are 0. The hidden layer (sometimes also called a projection layer) consists of 300 nodes and when trained, represents the actual word embeddings. Each node in the output layer is associated with a word in the vocabulary and outputs the probability that the given word appears in the context of the input word.

Formally, the objective of the Skip-gram model is to maximize:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log(p(w_{t+j}|w_t)) \tag{3.5}$$

where T is the size of the training data and c determines the size size of the context window. Probability of two words occurring in the same context is defined as:

$$p(w_{t+j}|w_t) = \frac{exp({v'_{w_{t+j}}}^{\mathsf{T}} v_{w_t})}{\sum_{w=1}^{V} exp({v'_w}^{\mathsf{T}} v_{w_t})} \tag{3.6}$$

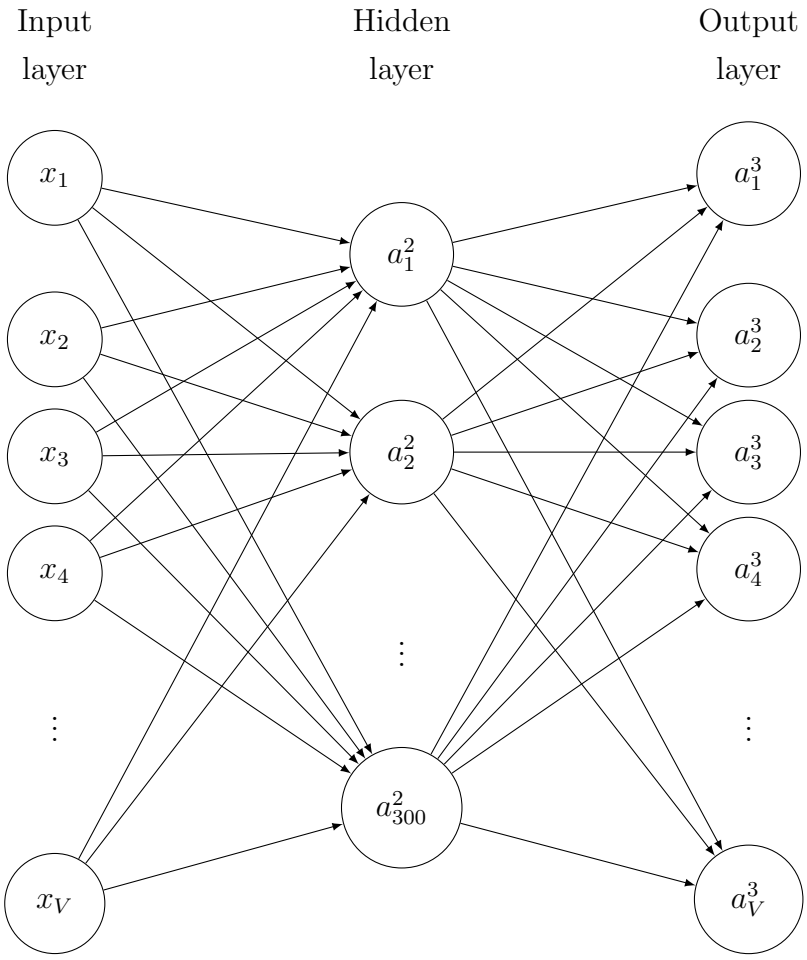


Figure 3.1: word2vec neural network architecture; V - size of the vocabulary; $x_{1..V}$ - (one-hot encoded) input word vector; $a^2_{1..300}$ - input word embedding; a_i^3 probability of co-occurrence of the word w_i and the input word

where v_w is the vector representation of word w (i.e. the output of the hidden layer), v'_w is the vector of weights associated with the node w in the output layer and V is the size of the vocabulary [Mikolov et al., 2013b, pp. 2-3].

The parameters of the neural network can be represented by two matrices: $\Theta^1 \in \mathbb{R}^{m \times V}$ for the weights needed for transition from the input (1) to the hidden (2) layer and $\Theta^2 \in \mathbb{R}^{V \times m}$ for the transition from the hidden (2) to the output (3) layer. There is no activation function between layer 1 and 2. This means the activation of the hidden layer is given simply by $a^2 = \theta^1 x$; x is the one-hot encoded input vector. Since only a single value in x is equal to 1 and all other values are zero, the activation of layer 2 for word w_i corresponds to $\Theta^1_{*,i}$; i.e. to the i^{th} column of the parameter matrix. Therefore, by training the neural network so that it outputs similar probabilities for words occurring in similar contexts (3.6), it learns to represent such words with similar vectors.

There are other ways to train word embeddings. Levy and Goldberg [2014] show the effect of choosing different types of contexts on the nature of similarity among the word vectors. As a response to the 'linear bag of words contexts' used by word2vec, they use 'dependency-based contexts' that take into account not only nearby words but also their dependency relationship to the input word. The authors conclude that the embeddings based on the latter type of context "are less topical and exhibit more functional similarity than the original Skip-gram embeddings". [Levy and Goldberg, 2014, p. 302] In word2vec type embeddings, among most similar words to word *florida* are words like *fla, alabama, gainesville, tallahassee* while vectors trained on dependency-based contexts favor words like *texas, lousiana, georgia, california.* [Levy and Goldberg, 2014, p. 305]

However, implementing and experimenting with architectures for extracting word vectors was out of the scope of the thesis. Instead, we choose to use the pre-trained 300-dimensional word vectors provided by Google[6]. The data contain 3 000 000 embeddings and were trained on about 100 billion words.

3.2.5 Language model prediction

label: Lm

Finally, a separate language model is trained on the 1 billion word benchmark dataset and used to produce its own predictions for the given context. These predictions (suggestions) were used as a single feature in some of the experiments described later.

Language model

Statistical language model is a model that assigns probability to a sequence of units. For this project the units correspond to words:

$$P(w_1, w_2, w_3, \dots w_t) = \prod_{i=1}^{t} P(w_i | w_1, w_2, w_3, \dots, w_{i-1}) \tag{3.7}$$

Since estimating such probability directly from data would be impossible due to the data sparsity, the above is further simplified by using n^{th} order Markov property:

$$P(w_1, w_2, w_3, \dots w_t) \approx \prod_{i=1}^{t} P(w_i | w_{i-(n-1)}, w_{i-(n-2)} \dots, w_{i-1}). \tag{3.8}$$

[6] `https://code.google.com/archive/p/word2vec/`

The sequence $w_{t-(n-1)}, w_{t-(n-2)} \ldots, w_{t-1}, w_t$ is called an n-gram. Using maximum likelihood estimate given a training dataset, the probabilities are estimated by

$$P(w_t|w_{t-(n-1)}, w_{t-(n-2)} \ldots, w_{t-1}) = \frac{\#(w_{t-(n-1)}, w_{t-(n-2)} \ldots, w_{t-1}, w_t)}{\#(w_{t-(n-1)}, w_{t-(n-2)} \ldots, w_{t-1})}, \quad (3.9)$$

where $\#(x)$ stands for the number of times the sequence x was observed in the data. Even with the simplifying assumption (3.8), the problem of data sparsity remains. When predicting the probability on some unseen data, some of the observed n-grams are likely to have no corresponding counts in the training data, making (3.9) unusable. To mitigate the problem couple of smoothing approaches have been designed. Generally, the approaches reserve some of the empirical probability mass for unseen examples. One such approach known as Kneser-Ney smoothing was used for the language model trained as part of this thesis.

Kneser-Ney interpolated smoothing

Chen and Goodman [1999] motivate this approach by the simple *San Francisco* example. If the bigram *San Francisco* occurs frequently in the training text, the unigram probability of the word *Francisco* will be relatively high. When other smoothing methods use this unigram probability as a back-off for higher order n-grams, they produce a large probability estimate for the following bigram: (-OOV-, Francisco), where -OOV- stands for any word not seen in the training set. However, since *Francisco* occurs only with *San*, the probability of the given bigram should be low.

Thus, instead of a simple unigram probability, Kneser-Ney smoothing uses the probability of the word occurring within a novel context:

$$p_{KN}(w_t) = \frac{|\{w_i : \#(w_i, w_t) > 0\}|}{|\{w_i, w_j : \#(w_i, w_j) > 0\}|}. \quad (3.10)$$

Setting h_t to represent the (n-1)-long history of a given word w_t, i.e. $h := w_{t-(n-1)}$, $w_{t-(n-2)} \ldots, w_{t-1}$, the general n-gram interpolated estimate of the probability is defined as:

$$p_{KN}(w_t|h_t) = \frac{\max(\#(h_t, w_t) - \delta, 0)}{\#(h_t)} + \lambda_{w_{t-1}} p_{KN}(w_{t-1}|h_{t-1}). \quad (3.11)$$

Here, δ is a constant standing for the strength of smoothing and λ is a normalizing constant set so that the estimated distribution sums up to 1:

$$\lambda_{w_{t-1}} = \frac{\delta}{\#(h_t)} |\{w_i : \#(w_{t-1}, w_i) > 0\}|. \quad (3.12)$$

Language model training and evaluation

For training the model, an implementation of the above algorithm by Heafield et al. [2013] was used. The 5-gram model was trained on the training portion of the 1 billion word benchmark and evaluated on a portion of the held-out data as described by Chelba et al. [2013] (Section 3.1.3). The data was preprocessed by merging the two forms of the indefinite article into *a/an* and replacing all numbers by the <*number*> token. The model is then evaluated by perplexity defined as:

$$2^{-\frac{1}{N}\sum_{t=1}^{N}\log_2 p_{KN}(x_t)}, \tag{3.13}$$

where N stand for the number of words in the test dataset. The resulting perplexity 70 is close to the result reported by a comparative study of different language models by Józefowicz et al. [2016], – 67.6.

Language model prediction as a feature

To use the trained language model as a feature for a classifier, the following process was employed: For each noun phrase, three candidate sentences were created by inserting *the* or *a/an* to the left boundary of the noun phrase, or leaving the space blank. The three candidates were then evaluated by perplexity (3.13) and the one with the lowest perplexity was selected as the winner. Its category, i.e. definite/indefinite/zero, was the new value of the feature for the given noun phrase. Furthermore, the suggestion was stored and used as part of the sentence when predicting the article for the following noun phrase.

3.3 Feature post-processing

Not all the features described in the previous sections can be used for classification directly. The categorical features need to be converted into numerical values. Firstly, the cut-off limit of 5 is applied so that all feature values that occur with less than the given threshold are discarder. This step leads to substantial reduction of the dimensionality of the feature space: considering all the features described in the previous section (and the post-processing steps described below), the number of features drops from 287 762 to 35 516. The price for this memory saving is a slight decrease in accuracy (see Section 4.1.3)

Firstly, each feature whose whose original values are strings (such as word forms or tags) is converted into n binary features, where n is the number of distinct feature values (see one-hot encoded vector in Section 3.2.4).

In a similar manner, features whose values are sets of strings, such as *words before head* in the EXTENDED feature set, are converted into series of binary fea-

tures so that each distinct value in a list forms a new binary feature (bag-of-words approach). Thus an example noun phrase *broad market averages*, with a *words-before-head* feature value "*broad market*", is represented by features *words-before-head-broad* and *words-before-head-market*, whose value is 1, and other feautures (*words-before-head-*{*personal/his/decline/...*}), which are set to 0.

Finally, for word embeddings vectors, each dimension of the vector is taken as a separate new feature.

4. Evaluation

In this chapter, the experiments conducted for this thesis are described and evaluated. The first two sections deal with classifiers based on the two machine learning algorithms described in Chapter 2. All the classifiers have been built and evaluated on the Penn Treebank Wall Street Journal data (described in more detail in Section 3.1.1), using the features presented in Section 3.2. The next section provides comparison of the classifier-based approach with the performance of a language model. Finally, for reference, the human performance on the task is given and compared to the best automatic method.

4.1 Logistic regression models

As already mentioned, the starting point for the experiments was the attempt to replicate the results given by Lee [2004]. The author explicitly mentions the use of the maximum entropy framework as described by Ratnaparkhi [1998], discussed in Section 2.1.2. However, implementing the algorithm directly was not part of this work; instead, logistic regression as provided by the *scikit-learn* package [Pedregosa et al., 2011] for the Python programming language was used. The change from maximum entropy to logistic regression model is justified in Chapter 2, specifically, sections 2.1.2 and 2.1.3 describe the equivalence of the models under conditions that — at least in the case of the replication experiment — were fulfilled.

4.1.1 Hyperparameter tuning

The only hyperparameter tuned for logistic regression models is the regularization parameter λ determining the strength of the penalty for learning high values of the model parameters θ. (See (2.13) and (2.15) for representation of logistic regression model and its cost function.)

The optimal value of this parameter is likely to differ for different models based on the dimensionality of the feature space, the amount of training data and the particular approach to multinomial classification (i.e. softmax formulation vs. one-vs-all vs. one-vs-one). Therefore, for each model trained and presented below, the optimal value was first estimated on the training data by 5-fold cross-validation.

The employed software implementation of logistic regression uses the parameter C that is defined as the inverse of λ, $C = \frac{1}{\lambda}$. For convenience, both values will be provided in the following reports.

4.1.2 Replication of a reported experiment

The replication of the article generation experiment reported by Lee [2004] was attempted by following its experimental setup, namely the choice and processing of the data, employed explanatory variables and the choice of the machine learning algorithm. A multinomial logistic regression model was fit on the training and tested on the test data described in Section 3.1.1. The only difference from the referenced section was the fact that for this experiment, discarding rare values of categorical features was not performed as there is no mention of such a step in the original article.

The resulting accuracy as compared to the baseline model (predicting the most frequent category to all examples) and the result reported by Lee [2004] is given in Table 4.1. As indicated by the results, the replication experiment failed

model	C (λ)	feature sets	accuracy
Baseline	—	—	71.1%
Multinomial LR	0.4 (2.5)	Orig	88.36%
Reported result	—	Orig	87.7%

Table 4.1: Evaluation of the replication experiment on the *test* data. *Baseline* stands for simple major vote, i.e. predicting no (zero) article for all test examples; *Multinomial LR* stands multinomial logistic regression model implemented as part of the thesis; and *Reported result* stands for the model by Lee [2004]. The second column indicates the regularization parameter used for training. The accuracy scores measure the performance on the *test* dataset.

to exactly meet the target accuracy. The difference between the two results is about half a percent. It is not clear why this discrepancy occurred.[1]

4.1.3 Cut-off value for categorical variables

The effect of trimming the low frequency values of categorical explanatory variables is investigated by training three separate multinomial logistic regression models for the cut-off values of $0, 3$ and 5. A cut-off value represents the threshold on frequency a feature value needs to exceed in order to be included in the data. Since categorical features are one-hot encoded, introducing the threshold leads to significant reduction in the number of final features.

The models are evaluated on the held-out data and the results, as well as the final number of features used by the models, are presented in Table 4.2.

[1] One attempt to get closer to the original result was disabling regularization as it is not mentioned in the paper. This leads to the accuracy of 87.93% on the test set.

Similarly to the previous experiment, the models use only the original set of features suggested by Lee [2004] (Section 3.2.1).

model	C (λ)	feature sets	threshold	accuracy	#features
Multinomial LR	0.4 (2.5)	ORIG	0	87.90%	81 169
Multinomial LR	0.4 (2.5)	ORIG	3	87.91%	25 237
Multinomial LR	0.6 (1.7)	ORIG	5	87.81%	18 872

Table 4.2: Evaluation of the effect of the threshold for cutting off low-frequency feature values on the accuracy and the number of final features. The models are compared on the *held-out* data.

Interestingly, setting a higher threshold can actually improve the performance of the model. The effect of increasing the threshold can be understood as a kind of "targeted regularization". Removing all the categorical values that occur less then 4 times is equivalent to setting their corresponding parameters θ to zero, which is what the regularization term approaches for all the features when it increases. Thus, introducing a threshold can be seen as a means of preventing overfitting. However, by rising the threshold higher the bias introduced to the model becomes too high and the performance starts decreasing. As described in Section 3.3 the threshold of 5 was accepted for further experiments due to the reduction of number of features. This advantage becomes even more relevant when training gradient boosted trees.

4.1.4 Features

Next, the effect of the newly designed features on the performance of the classifier is evaluated. At first, a model is trained on each of the new feature set together with the original feature set. Then the feature sets are added together iteratively and the performance is measured with respect to the growing number of features. Finally the model utilizing all the available features is trained. The corresponding results are presented in Table 4.3.

The results show that when taken independently, each feature set presented in Section 3.2 brings some new useful information for the model. Specifically, the biggest improvement is achieved by the *'extended'* feature set, which leaves behind all the features that utilize additional sources of data. In that feature set, the major role is played by the modification of the six list features: i.e. representing the context of the noun phrase head as a fixed string rather than a bag of words (or tags). Interestingly, utilizing each dimension of a pre-trained word embeddings as a separate explanatory variable can also improve performance. Other two

model	C (λ)	feature sets	accuracy
Multinomial LR	0.4 (2.5)	Orig	87.81%
Multinomial LR	0.6 (1.7)	Orig-Ext	88.39%
Multinomial LR	0.4 (2.5)	Orig-Cnt	88.05%
Multinomial LR	0.4 (2.5)	Orig-Emb	88.27%
Multinomial LR	0.4 (2.5)	Orig-Lm	88.08%
Multinomial LR	0.2 (5.0)	Orig-Ext-Cnt	88.33%
Multinomial LR	0.2 (5.0)	Orig-Ext-Cnt-Emb	88.80%
Multinomial LR	0.3 (3.3)	Orig-Ext-Cnt-Emb-Lm	89.08%

Table 4.3: Evaluation of models trained on different sets of features. The accuracy values represent the performance on the *held-out* data.

features, namely a countability prediction based on British National Corpus and a language model trained on another large corpus bring less improvement. Since both features utilize the same type of context as used by the original feature set, it might be the case that most of the new useful information is already captured by other variables.

4.1.5 Approaches to multinomial classification

The next set of experiments compares the approaches to multinomial classification as discussed in Section 2.1.3. Namely, multinomial logistic regression — which has been used so far — and two methods of combining binary classifiers: one-vs-one and one-vs-rest approach. For one-vs-one approach the method of pairwise coupling was implemented and used. For one-vs-rest approach two types of models were trained: an 'out-of-box' method provided by the employed software package and a manually trained and combined set of binary classifiers. In contrast to the manual method, the 'out-of-box' method does not enable tuning the individual classifiers separately as it exposes only one set of hyperparameters for all the binary models. By considering the last two methods separately, an effect of tuning the binary classifiers individually can be estimated. All models were trained with all available features. The evaluation of the models is given in Table 4.4.

The results show that for the given problem, training binary classifiers independently is a better approach than fitting all the parameters jointly as is the case in the multinomial logistic regression model. Out of the two methods of combining binary classifiers, the one-vs-rest method outperforms the one-vs-one method with pairwise coupling. Finally, the experiment shows a minor improvement that can be achieved by tuning the binary classifiers independently as opposed to us-

model	C (λ)	accuracy
Multinomial LR	0.3 (3.3)	89.08%
One-vs-one	a|0:1.25 (0.8); a|the:0.4 (2.5); the|0:0.7 (1.43)	89.33%
One-vs-rest oob	0.6 (1.7)	89.45%
One-vs-rest man	a:0.6 (1.7); the:0.5 (2); 0:0.7 (1.43);	89.49%

Table 4.4: Evaluation of four different approaches to multi-class classification using logistic regression models. The models were trained using all the available features and the scores were measured on the *held-out* data. The middle column shows the regularization parameter. Binary classifiers are specified as 'x|y' or 'x' meaning x-vs-y and x-vs-rest, respectively.

ing the available 'out-of-box' implementation that uses a single regularization parameter for all the models.

The differences between the approaches are further illustrated by Figure 4.1, where the performance of the models with respect to the size of the training data is compared. Four different subsets of the training dataset were randomly selected with sizes: 20 000, 50 000, 100 000 and 200 000 examples. The dataset consisting of all available training examples (263 088) was also added. Again, the value of the regularization parameter was estimated for each model and each training dataset using cross-validation.

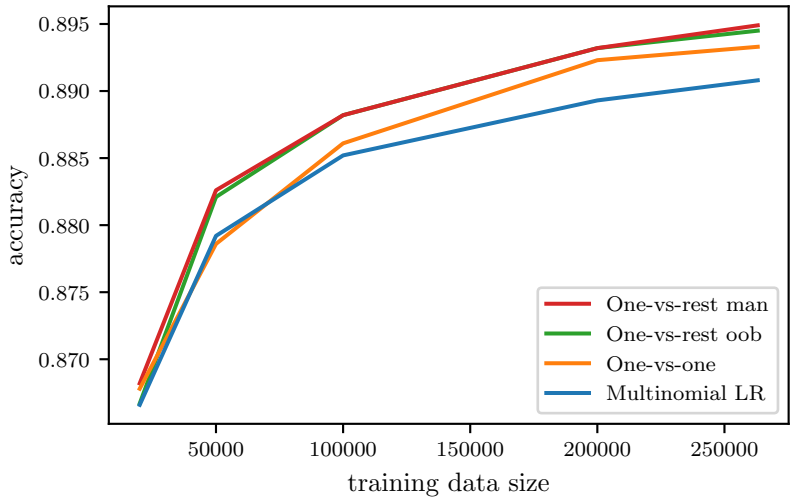


Figure 4.1: Performance of different approaches to multi-class classification using logistic regression models as a function of the training data size. The models use all available features and were tuned for each dataset separately.

The results show that the multinomial logistic regression model was outperformed systematically by the other approaches. The 'out-of-box' and 'manual'

approaches towards the one-vs-rest classification are very close to each other regardless of the data size. Despite their closeness, the 'out-of-box' model did not achieve higher score than its alternative in any experiment. Finally, the one-vs-one approach starts being more effective (with respect to multinomial logistic regression) only when trained on larger datasets. This is in accord with the disadvantage of this approach mentioned in Section 2.1.3, namely the fact that the individual binary models cannot use all the provided data.

4.1.6 The best model

Out of the experiments described above, the best accuracy on the held-out data was achieved by a classifier composed of three binary models grouped together in the one-vs-rest fashion. Each classifier was tuned individually, i.e. the strength of its regularization term was estimated independently. The model uses all available features. Its accuracy measured on the held-out dataset is 89.49%. On the **test dataset** it achieves the accuracy of **90.00%**.

4.2 Gradient boosted trees

In what follows, attempts on improving the achieved results by means of a different machine learning algorithm are described. As mentioned in Section 2.2, a specific implementation of the gradient boosted trees algorithm known as *XGBoost* [Chen and Guestrin, 2016] was used. While the algorithm achieves state-of-the-art results on many tasks, it is also significantly more difficult to tune (compared to logistic regression) as there are many hyperparameters influencing the final performance. After introducing the parameters and their relationship to the model, the selected approach to their tuning is described together with the final results.

4.2.1 Hyperparameter tuning

For tuning the gradient boosted trees model the following hyperparameters were investigated:

- **Objective Function**
  As explained in Section 2.2, XGBoost implementation is able to handle different types of objective functions. For this experiment, the predefined softmax function (2.15) was chosen.

- **Number of trees**
  The actual amount of weak learners in the resulting ensemble. The learning

algorithm assumes a fixed number of estimators, so it must be given by the researcher.

- **Maximum tree depth**
  An upper limit on the number of tree levels. Tree depth controls the order of interactions between variables within the model. Setting this parameter high results in more complicated and strong learners, which can lead to over-fitting. The minimum depth of 1 results in trees with one split only and consequently no interaction between variables. Such a tree is sometimes called a 'decision stub'.

- **Minimum weight of a child**
  By setting a requirement on the minimum weight of a child, the algorithm will only split a node when the weights of both new leaves are higher than the given threshold. Higher values result in less complicated trees.

- **Gamma**
  The gamma parameter of the regularization term as shown in (2.36). It represents another way of preventing the algorithm from learning too complicated base learners. Each split in the tree must result in cost reduction that is higher than the given value.

- **Column subsampling**
  Column subsampling is a method often used for random forests. The parameter represents a proportion of features used for building a tree. For each tree a sample of the corresponding size is randomly drawn from the feature pool. The parameter value of 1 represents no subsampling. Randomly limiting the number of features a tree can use results in a collection of trees that are less correlated. Less correlated trees can better reduce the variance of the model, i.e. limit the amount of over-fitting.

- **Row subsampling**
  Similarly to the previous parameter, when setting the parameter value to be less then 1, a random subsample of the training examples is drawn from the training data. Again, this is done for each new tree in the ensemble. The motivation is the same as in the case of column subsampling.

- **Shrinkage**
  Shrinkage can be seen as an analogy to learning rate. It takes a value from 0 to 1 which is then used to scale down the importance of each new tree. This "leaves space for future trees to improve the model." [Chen and Guestrin, 2016, p. 3]

Tuning process

As mentioned above, the *objective function* was set to *softmax* to perform multinomial classification. For selecting the values for the following parameters, 5-fold cross-validation on the training data was used to get the performance of the model

Further, in accord with [Hastie et al., 2009, p. 365], the *shrinkage* parameter was set to a low value (0.1) and the *number of trees* was estimated. In this step, the performance of the model is measured after a tree is added to the ensemble. If there is no improvement of the performance in the course of the last 20 trees, the algorithm finishes and the number of trees corresponding to the best performance is selected. During this step, the other parameters were set to their default values: *maximum tree depth* – 5, *minimum weight of a child* – 1, *gamma*: 0, *column subsampling*: 0.8, *row subsampling*: 0.8.

Then optimal values for *maximum tree depth* and *minimum weight of a child* were estimated by performing a grid-search over the values (5, 7, 9, 11, 13) for the former and (1, 3, 5) for the latter parameter.

In the next step the *gamma* parameter was selected out of eight possible values: (0, 0.1, 0.2 . . . 0.7)

Finally *column subsampling* and *row sumbsampling* were selected using grid-search over the values (0.7, 0.8, 0.9, 1) for both of the parameters.

4.2.2 The best model

Performing the parameter tuning in the way described in the previous section resulted in the following set of parameters:

- *Objective Function*: softmax
- *Number of trees*: 1 531
- *maximum tree depth*: 11
- *minimum weight of a child*: 1
- *gamma*: 0.6
- *column subsampling*: 0.9
- *row subsampling*: 0.9
- *Shrinkage*: 0.1

With the given set of values the model achieves 91.86% accuracy on the held-out data set and **91.84%** accuracy on the **test data**. Table 4.5 summarizes the most important models learned so far.

model	feature sets	accuracy
Baseline	—	71.1%
Reported result	ORIG	87.7%
Multinomial LR	ORIG	88.36%
One-vs-rest oob	ORIG-EXT-CNT-EMB-LM	89.08%
Boosted trees	ORIG-EXT-CNT-EMB-LM	**91.84%**

Table 4.5: Evaluation of the best logistic regression and gradient boosting models. For reference, the baseline accuracy is given followed by the result reported by Lee [2004] and the model attempting to replicate the result. The accuracy values represent the performance on the *test* data.

By changing the machine learning algorithm from logistic regression to gradient boosted trees, the accuracy on the test data increased by about 2.8%. This represents about 25% reduction in error.

Comparison between the best logistic regression model and the gradient boosted trees classifier is further illustrated by Figure 4.2. Both models are evaluated on different portions of training and held-out data (20 000, 50 000, 100 000, 200 000 and 263 088 examples). Again, each model was specifically tuned for the given training data size. Gradient boosted trees outperform the logistic regression model on all the datasets for the given problem. Moreover, its training accuracy is very high as well as the difference from the test accuracy. This can be understood as indication of over-fitting the training data despite the parameter tuning described above.

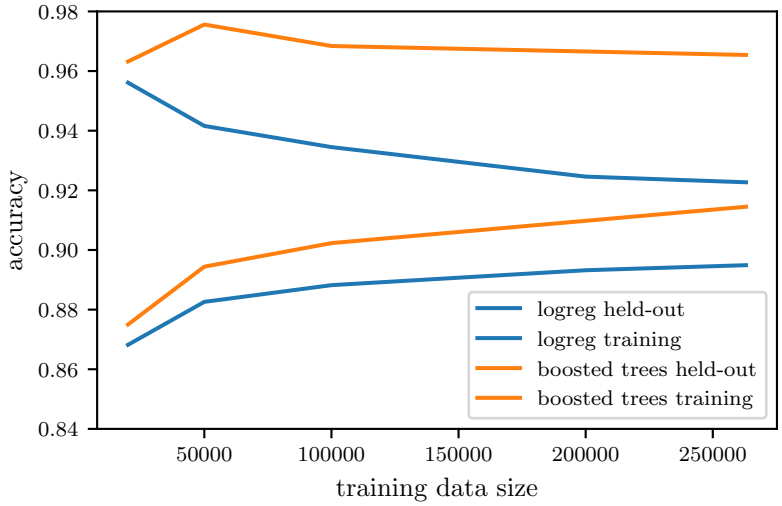


Figure 4.2: Training and held-out data accuracy for logistic regression and boosted trees models

4.3 Language model

The main approach presented in the thesis is linguistically motivated in the sense that it first predicts the linguistic structure of each sentence (by part-of-speech tagging and parsing) and only then it assigns specific elements of such a structure with an article. Although this seems analogous to what speakers of English do, it also raises some problems. Namely, the assumption of having correct parses is hard to guarantee in practice.

In this section, an alternative approach is evaluated: each position between two words is considered as a potential place for an article. For each such a position three options corresponding to three article forms (*zero*, *the*, *a/an*) are evaluated by means of a language model. The winning alternative is kept and the next position is evaluated until the end of the sentence is reached. The process is illustrated in Table 4.6.

step	candidate string	cost
1	[—] despite attempts to prevent this, ...	501
1	[**a/an**] despite attempts to prevent this, ...	875
1	[**the**] despite attempts to prevent this, ...	1086
2	despite [—] attempts to prevent this, ...	501
2	despite [**a/an**] attempts to prevent this, ...	590
2	despite [**the**] attempts to prevent this, ...	430
3	despite the attempts [—] to prevent this, ...	430
3	despite the attempts [**a/an**] to prevent this, ...	681
3	despite the attempts [**the**] to prevent this, ...	689

Table 4.6: Illustration of the process of generating articles by a language model. For each step (position between words) three candidates are generated and evaluated by the model.

The cost of each candidate is perplexity (3.13) (explained in Section 3.2.5). The language model is identical to the one described in the same section.

With this approach, it is not desirable to rely on parsed data for evaluation, as was the case in the previous experiments. The articles can now be predicted in any position within a sentence, so the accuracy of noun phrases is no longer relevant. Instead, the performance is measured by number of errors per 100 words. From the perspective of the model, an error can be either an *insertion* (predicting an article for an originally empty position) *substitution* (predicting a wrong article for the given position) or *deletion* (wrongly predicting no article for the given position). To find the number of errors, corresponding sentences from

the original and predicted texts are tokenized and compared. The 'a/an' token is considered to be equal to both 'a' and 'an'. The results on the test portion of the Penn Treebank are given in Table 4.7. For reference, the logistic regression and gradient boosting models are evaluated in the same manner. Both models used automatic parses for the prediction.

model	errors/100 words	#correct	#swaps	#insertions	#deletions
LM	19.73	3 340	732	10 427	51
LM-tun	7.24	19	8	8	4096
Logreg	6.80	452	328	193	3343
Boosting	6.87	232	185	12	3706

Table 4.7: Evaluation of the language model (LM) with respect to the logistic regression (Logreg) and gradient boosting models (Boosting). The models are evaluated on the Penn Treebank *test* data consisting of about 57 000 words. The main metric is the number of errors per 100 words. #correct represents number of correctly predicted articles (definite or indefinite, zero is not included). The other columns represent the distribution of the error types.

Without any modification, the proposed method of using a language model for article prediction did not work. The main source of errors is the overuse of the indefinite article as illustrated by a random sentence taken from the test data:

> Original " I 'd like to see that initiative , and i have n't . a/an " everybody thouth we were looking at "
>
> LM **a/an** " I 'd like to see that **the** initiative, and I have **a/an** n't .

The logistic regression and gradient boosting models, on the other hand, seem to be too conservative, as they attempt a prediction quite rarely. This is demonstrated by their main source of error, i.e. deletion. Interestingly, contrary to the evaluation of the classification task, here the gradient boosting model performs worse than logistic regression.

In order to compensate for the over-generating of the language model, two parameters were introduced: *perplexity threshold* and *decision margin*. Both parameters were supposed to restrict the language model to only such predictions where it is 'confident enough'. *Perplexity threshold* sets the limit in absolute terms by setting a value above which a decision is not considered. *Decision margin* attempts to control the confidence relatively, i.e. it sets the minimum relative difference in the cost between the first and second best candidates for the

given position. However, when both parameters were tuned by grid-search on the held-out dataset, the solution gravitated towards the baseline performance of not predicting anything as seen by the results of the *LM-tun* model in Table 4.7.

4.4 Human performance

To put the results from the previous sections into perspective, 4 expert translators were asked to perform the same article generation task. The text they were given was composed by selecting two random files from the British National Corpus (see Section 3.1.2) and extracting three consecutive paragraphs from each, while discarding all the articles. Although this setting produces only two distinct pieces of texts, it attempts to mimic a real-world scenario by creating excerpts that are large enough to capture some long-range dependencies among the noun phrases. Thus under the constrained resources, it can be viewed as a trade-off between producing statistically more relevant results on the one hand and more realistic tasks on the other. Both the original text as well as the annotators' responses are given as attachments in Section 4.5.

The generated text consists of 640 words. The accuracy was measured by first manually counting the number of positions an article could have occurred. Then the number of differences in articles between the original and the predicted text was computed. Table 4.8 compares the performance of the four annotators.

annotator	#errors	#swaps	#insertions	#deletions	accuracy
A	29	4	4	21	82%
B	45	7	13	25	72%
C	22	2	6	14	87%
D	17	7	3	7	90%
mean	28.25	5	6.5	16.75	82.75%

Table 4.8: Human performance on the article generation task. Measured on text with 163 candidate noun phrases.

The experiment revealed quite large differences between the annotators. In Chapter 1, it was mentioned that in some situations there might be more options to express the notion of definiteness of the given noun phrase. The results show that this might happen quite often. The following example illustrates such a case:

> Original In a monopoly situation, **the producer** may be able to use his market power at the expense of the consumer, ...

Annotators A, D In a monopoly situation, **a producer** may be able to use his market power at the expense of the consumer, ...

Here, the noun *producer* can be seen as having definite associative reference (i.e. as being defined based on the association with the concept of *the market:* {*consumer, producer, demand, supply, ...*}, which has already been mentioned at this point), or it can be seen as having singulative indefinite reference (i.e. *any producer/some producers*). A different type of ambiguity is illustrated by another example:

Original High prices were maintained from 1974 onwards in the face of inelastic demand for oil from oil-importing countries, but **an oil glut** in the 1980s put pressure on individual countries to ...

Annotators B, D ..., but **the oil glut** in the 1980s put pressure on individual countries to ...

By choosing the first option, one interprets the *oil glut* as one in a sequence of such periods of oversupply, which happened to occur in the 1980s. The other option suggests the *glut of the 1980s* is a particular period known to the reader. The latter interpretation leaves no place for other such gluts during the 1980s. Apparently, to predict (or correct) an article in certain cases requires very deep understanding of the intended meaning and cannot be predicted from the text alone.

4.4.1 Comparison to automatic methods

Due to the different type of data, the accuracy given in Table 4.8 should not be directly compared to the results reported in the sections on machine learning experiments. Table 4.9 illustrates the difference in the distributions of the article types over the two datasets. The three categories are more evenly distributed in the manual test data. While the relative portion of the indefinite articles remains comparable, the main change happens in the definite and zero article categories.

To facilitate the comparison between the performance of manual annotators and automatic methods of predicting the articles, the models discussed in the previous sections were evaluated on the same dataset. The results are shown in Table 4.10

Clearly, machine learning models learned on the Penn Treebank do not generalize well to this new context. They remain overly conservative and their performance is very close to the baseline. For 68 real articles, they predicted only 4, one of which was false.

	penn-test	manual-test
zero	71.1%	59%
the	19.7%	29%
a/an	9.3%	12%

Table 4.9: Distribution of the articles (determiners) over the Penn Treebank and BNC manual test data

	errors/100 words	#correct	#swaps	#insertions	#deletions
Man-best	2.7	54	7	3	7
Man-avg	4.4	46	5	7	17
Logreg	10.2	3	1	0	64
Boosting	10.2	3	1	0	64
Baseline	10.6	0	0	0	68

Table 4.10: Comparison of manual annotation (Man-best, Man-avg) to automatic methods (Logistic regression, gradient boosting) on 6-paragraph BNC data of 640 words. The main metric is the number of errors per 100 words. #correct represents number of correctly predicted articles (definite or indefinite, zero is not included). The other columns represent the distribution of the error types.

Comparing the performance of manual and automatic annotation can also be approached from the other direction. Instead of evaluating the automatic models on the dataset used for manual annotation, one can estimate the performance of the annotators on the dataset used for the evaluation of the machine learning models.

Firstly, a confusion matrix is obtained from the manually annotated data. This is possible because the number of noun phrases, which corresponds to the number of potential articles, has already been extracted (also manually). The confusion matrix for the best annotator is given in Table 4.11.

	the	a/an	zero
the	35	6	7
a/an	1	19	0
zero	2	1	92

Table 4.11: Confusion matrix for the most successful manual annotation. The rows correspond to the articles found in the original text, the columns correspond to the predictions.

Out of each row of the matrix, the probability distribution over the predic-

tions given the true article was estimated by maximum likelihood estimation: $p(x|y) = \frac{\#\{x \ \& \ y\}}{\#y}$ (x and y denote the predicted and true article, respectively).

The three probability distributions are then used to estimate the distribution of errors that would be made on the Penn test set. Specifically, let $c_{x|y}$ denote the number of times x is predicted for any noun phrase with a real article y on that dataset. Then, the distribution of $c_{x|y}$ is estimated as $\hat{c}_{x|y} = c_y p(x|y)$, where c_y is defined as the number of noun phrases whose true class is y. By computing $\hat{c}_{x|y}$ for all the nine combinations of x and y, one creates an estimated confusion matrix on the new dataset. From this matrix, the final estimate of the accuracy can be expressed as $\frac{\sum_x \hat{c}_{x|x}}{\sum_x \sum_y \hat{c}_{x|y}}$.

Table 4.12 gives the estimated accuracies for the best and the average human performance together with the empirical accuracies of the best gradient boosting and logistic regression models. The results show that when the models are tested on the data they were prepared for, their comparison to human performance is much more favorable. Both models perform slightly worse than the best human annotator and above the average annotator performance.

model	accuracy
Best annotator (estimated)	91.96%
Boosted trees	91.84%
Logistic regression	89.08%
Average annotator (estimated)	85.98%

Table 4.12: Comparison of manual annotation to automatic methods on the Penn Treebank test set. The accuracies of the manual performance are estimated from the error distribution from another dataset.

Of course, the above conclusion is valid only if the estimate of the manual performance on the dataset is reliable. There are two issues with this assumption. Firstly, the estimates of the three probability distributions $p_y(x) = p(x|y)$ are based on limited data. Namely, p_{the} is based on 48, $p_{\text{a/an}}$ on 20 and p_{zero} on 95 observations. Secondly, even if taking p_y for granted, the accuracy estimation assumes that for a given article, the distribution of its prediction errors is the same for both the manual and automatic dataset. In other words, it assumes there is the same level of uncertainty connected with an article no matter which dataset is used.

Conclusion

In accord with previous research [Minnen et al., 2000], [Lee, 2004], [Turner and Charniak, 2007], [Sun et al., 2015], the thesis interprets the problem of article error correction as an article generation task. Firstly, by looking into the previous literature on the subject, a group of papers was identified, which provided comparable results in terms of problem formulation, approach and the data used. Out of this group, the article reporting the best results was selected.

As the first step, the selected article was replicated (Section 4.1.2). A logistic regression model was trained on the training section of the Penn Treebank and evaluated on its test section. For each noun phrase, the model predicts one of the three possible categories corresponding to the article choice: *the* for the definite article, *a/an* for the indefinite articles and *zero* for noun phrases with no article. The features used are the ones described in the paper. This leads to the accuracy of 88.36% as compared to the original 87.7%. The source of this difference was not identified.

This baseline experiment was further improved in several ways. Firstly, new feature sets were designed that would hopefully improve the representation of the examples. The first set of features used only the information already available in the training data and achieved improvement from 87.81% to 88.39% as measured on the held-out dataset. Then features using additional data sources were added: prediction of countability category, prediction of a language model and word embeddings representation of the head of the noun phrase. When used together, the accuracy of the model further improved to 89.08% (Section 4.1.4).

Next, the effect of different approaches to extending binary logistic regression to multi-class problems was evaluated. Apart from the original multinomial logistic regression using the softmax function, one-vs-one and one-vs-rest methods of combining binary classifiers were evaluated. For the given problem, one-vs-rest approach provided the best result raising the held-out accuracy to 89.49% (Section 4.1.5).

The final attempt to improve the performance was made by replacing logistic regression with gradient boosted trees. After tuning the model and running it on the same data as the models discussed above, the accuracy of the classification improved to 91.86% for the held-out dataset. When measured on the test data, the performance is 91.84% (Section 4.2). This represents an increase of accuracy of 4.14% when compared to the best reported result on the task known to the author – 87.7% [Lee, 2004]. This corresponds to about 34% reduction in error.

An alternative approach based solely on the predictions of a large language

model was also investigated. However, it was not found useful when it was evaluated on the Penn Treebank test data (Section 4.3).

As a reference, the performance of human annotators on the same task is given (Section 4.4). Because the data were taken from another source, the comparison is not straightforward. It turned out that evaluating the trained model on the new dataset did not produce good results as the model was too conservative when predicting article changes. On the other, when the automatic and manual methods are compared on the dataset the models were prepared for, the performance of the best model matches the *estimated* performance of the best annotator.

Although this work presents an improvement on the best result for the given task by Lee [2004], there remains much to be done in terms of practical application of the suggested approach. The model was trained on newspaper articles focused on the world of finance, which suggests the generalization of such a model might be limited. This suggestion is also supported by one of the experiments.

In recent years, recurrent neural networks and long short-term memory neural networks in particular get much attention also in the NLP community [Mikolov et al., 2013b], [Józefowicz et al., 2016]. While gradient boosted trees seem to work well with the given problem, it would be interesting to see if the neural network based models can improve the performance by utilizing less constrained context of each noun phrase.

Bibliography

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN 9780412048418.

Chris Callison-Burch, Philipp Koehn, Philipp Monz, and Omar F. Zaidan, editors. *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Edinburgh, Scotland, July 2011. ISBN 9781937284121.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013. URL `http://arxiv.org/abs/1312.3005`.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL `http://arxiv.org/abs/1603.02754`.

Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2010.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. Building a large annotated corpus of learner english: The NUS corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Libuše Dušková, Dagmar Knittlová, Jaroslav Peprník, Zdeňka Strnadová, and Jarmila Tárnyiková. *Mlauvnice současné angličtiny na pozadí češtiny*. Academia, Praha, 2006. ISBN 9788020022110.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Jane E. Gressang. *A frequency and error analysis of the use of determiners, the relationships between noun phrases, and the structure of discourse in English essays by native English writers and native Chinese, Taiwanese, and Korean*

learners of English as a Second language. PhD thesis, The University of Iowa, 2010.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(02): 115–129, 2006.

Tevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York, 2009. ISBN 9780387848587.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, NIPS '97, pages 507–513, Cambridge, MA, USA, 1998. MIT Press.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, pages 5:1–5:9, New York, NY, USA, 2014. ACM.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics - Volume 2*, ACL '13, pages 690–696, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016. URL `http://arxiv.org/abs/1602.02410`.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (Second Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 9780131873216.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Kevin Knight and Ishwar Chander. Automated postediting of documents. In *Proceedings of the twelfth national conference on Artificial intelligence - Volume 1*,

AAAI '94, pages 779–784, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

John Lee. Automatic article restoration. In *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, HLT-SRWS '04, pages 31–36, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, 2014. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 9780521865715.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a. URL `http://arxiv.org/abs/1301.3781`.

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

George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Guido Minnen, Francis Bond, and Ann Copestake. Memory-based learning for article generation. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, CoNLL '00, pages 43–48, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Ryo Nagata, Takahiro Wakana, Fumito Masui, Atsuo Kawai, and Naoki Isu. Detecting article errors based on the mass count distinction. In *Proceedings of the Second international joint conference on Natural Language Processing*, IJCNLP'05, pages 815–826, Berlin, Heidelberg, 2005. Springer-Verlag.

Hwee Tou Ng, Mei Wu Siew, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Hwee Tou Ng, Mei Wu Siew, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, Baltimore, Maryland, 2014. Association for Computational Linguistics.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A Comprehensive grammar of the English language*. Longman, London, 1985. ISBN 9780582517349.

Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Philadelphia, PA, USA, 1998.

Alla Rozovskaya and Dan Roth. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*, 2010.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. The University of Illinois system in the CoNLL-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Chengjie Sun, Xiaoqiang Jin, Lei Lin, Yuming Zhao, and Xiaolong Wang. Convolutional neural networks for correcting english article errors. In *Natural Language Processing and Chinese Computing: 4th CCF Conference*, pages 102–110, Nanchang, China, 2015. Springer International Publishing.

Jennie Turner and Eugene Charniak. Language modeling for determiner selection. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

David Vadas and James R. Curran. Parsing internal noun phrase structure with Collins' models. In *Proceedings of the Australasian Language Technology Workshop 2007*, pages 109–116, Melbourne, Australia, December 2007.

List of Figures

List of Tables

Attachments

4.5 Manual annotation texts

4.5.1 Original text

Sell-through, as the retail sector of the video market is obscurely known, is at the moment very much a bull market, and there has lately been a growing shift to releasing more recent movies in this way. This has been capped by the well publicised current appearance of no less than Rain Man (Warner) for sale (£14.99) as well as for rent, when hitherto the two spheres have tacitly been regarded as mutually exclusive. But for all the changes, the prime emphasis in the retail market, so far as movies are concerned, still seems to be on the past, and predominantly on the mainstream Hollywood past.

Current refinements, no doubt with Christmas present buying in mind, include boxed sets of three or more related cassettes from Warners — for example, all three of James Dean's movies for £29.95, or for the same price, Marilyn Monroe in The Prince and the Showgirl, The Misfits and (a bit of a scoop, since it has not been on video before) Some Like It Hot. This kind of collection, though usually available individually, is increasingly and intelligently proving to be a marketing ploy. For random instance, CBS/Fox has recently released (£9.99 each) not only Cronenberg's 1986 The Fly and its far from negligible 1958 forerunner, but also the 1959 quick-buck sequel to the latter, Return Of The Fly, a collector's item, though not necessarily in qualitative terms.

And to move from B to Z movies, connoisseurs of the bizarre can now lay in their own copy of Edward Woods' Plan 9 From Outer Space (Palace, £14.99), once voted the most incompetent film of all time. The rental sector meanwhile provides — along with all the box-office successes which nowadays transfer to tape within a few months and probably need no further introduction — the chance to catch up on a variety of (often more deserving) movies which have been less widely seen in cinemas here.

Competitive markets are essential if the market system is to operate effectively. In a monopoly situation, the producer may be able to use his market power at the expense of the consumer, although the likelihood of this happening will be moderated if close substitutes for the product exist or if the monopolist fears entry of new, but similar, products into the market, attracted by high prices and the profits being made. It is relatively rare for a firm to have an absolute monopoly of the market, but there are many instances where a small number of

large firms dominate a market — this is called an oligopoly. Such firms may act overtly (and illegally) or covertly (still illegally, but hard to detect) as if they were in a monopoly situation.

Agreements or understandings could cover price-fixing and/or sharing out the available work without resorting to competition (which would have resulted in lower prices). In order for the agreement to stick, no single firm must break ranks, encouraged by the prospect of greater market share by lowering its price, or else a free-for-all may develop producing a competitive price, as if the market had been operating smoothly. The operation of such a price-fixing agreement can be seen in the way that the price of oil has been fixed by the Organisation of Petroleum Exporting Countries (OPEC) cartel.

High prices were maintained from 1974 onwards in the face of inelastic demand for oil from oil-importing countries, but an oil glut in the 1980s put pressure on individual countries to reduce their prices in order to gain market share. Despite attempts to prevent this by OPEC, the price of oil plummeted in 1986. Even if there are no formal agreements to interfere with the market, implicit understandings may be reached, in which case the behaviour of the firms involved might, in practice, be the same as if a formal agreement had existed.

4.5.2 Annotator A

Sell-through, as a retail sector of the video market is obscurely known, is at the moment very much a bull market, and there has lately been a growing shift to releasing more recent movies in this way. This has been capped by the well publicised current appearance of no less than Rain Man (Warner) for sale (£14.99) as well as for rent, when hitherto the two spheres have tacitly been regarded as mutually exclusive. But for all changes, a prime emphasis in the retail market, so far as movies are concerned, still seems to be on the past, and predominantly on the mainstream Hollywood past.

Current refinements, no doubt with Christmas present buying in mind, include boxed sets of three or more related cassettes from Warners — for example, all three of James Dean's movies for £29.95, or for same price, Marilyn Monroe in Prince and Showgirl, Misfits and (a bit of a scoop, since it has not been on video before) Some Like It Hot. This kind of collection, though usually available individually, is increasingly and intelligently proving to be a marketing ploy. For random instance, CBS/Fox has recently released (£9.99 each) not only Cronenberg's 1986 Fly and its far from negligible 1958 forerunner, but also the 1959 quick-buck sequel to the latter, Return Of Fly, collector's item, though not necessarily in qualitative terms.

And to move from B to Z movies, connoisseurs of the bizarre can now lay in their own copy of Edward Woods' Plan 9 From Outer Space (Palace, £14.99), once voted the most incompetent film of all time. Rental sector meanwhile provides — along with all box-office successes which nowadays transfer to tape within a few months and probably need no further introduction — the chance to catch up on a variety of (often more deserving) movies which have been less widely seen in cinemas here.

Competitive markets are essential if the market system is to operate effectively. In a monopoly situation, a producer may be able to use his market power at the expense of the consumer, although the likelihood of this happening will be moderated if close substitutes for the product exist or if the monopolist fears the entry of a new, but similar, products into market, attracted by high prices and profits being made. It is relatively rare for firm to have an absolute monopoly of market, but there are many instances where a small number of large firms dominate the market — this is called an oligopoly. Such firms may act overtly (and illegally) or covertly (still illegally, but hard to detect) as if they were in a monopoly situation.

Agreements or understandings could cover price-fixing and/or sharing out available work without resorting to competition (which would have resulted in lower prices). In order for agreement to stick, no single firm must break ranks, encouraged by the prospect of a greater market share by lowering its price, or else a free-for-all may develop producing competitive price, as if the market had been operating smoothly. The Operation of such a price-fixing agreement can be seen in way that price of oil has been fixed by the Organisation of Petroleum Exporting Countries (OPEC) cartel.

High prices were maintained from 1974 onwards in the face of inelastic demand for oil from oil-importing countries, but oil glut in 1980s put pressure on individual countries to reduce their prices in order to gain a market share. Despite attempts to prevent this by OPEC, the price of oil plummeted in 1986. Even if there are no formal agreements to interfere with the market, implicit understandings may be reached, in which case the behaviour of the firms involved might, in practice, be the same as if a formal agreement had existed.

4.5.3 Annotator B

Sell-through, as a retail sector of the video market is obscurely known, is at the moment very much bull market, and there has lately been a growing shift to releasing more recent movies in this way. This has been capped by well publicised current appearance of no less than Rain Man (the Warner) for sale (£14.99) as

well as for rent, when hitherto the two spheres have tacitly been regarded as mutually exclusive. But for all changes, the prime emphasis in the retail market, so far as movies are concerned, still seems to be on the past, and predominantly on the mainstream Hollywood past.

Current refinements, no doubt with Christmas present buying in mind, include boxed sets of three or more related cassettes from the Warners — for example, all three of the James Dean's movies for £29.95, or for the same price, Marilyn Monroe in the Prince and Showgirl, the Misfits and (a bit of a scoop, since it has not been on video before) the Some Like It Hot. This kind of collection, though usually available individually, is increasingly and intelligently proving to be a marketing ploy. For random instance, CBS/Fox has recently released (£9.99 each) not only the Cronenberg's 1986 Fly and its far from negligible the 1958 forerunner, but also the 1959 quick-buck sequel to the latter, the Return Of Fly, a collector's item, though not necessarily in qualitative terms.

And to move from B to Z movies, connoisseurs of bizarre can now lay in their own copy of the Edward Woods' Plan 9 From Outer Space (Palace, £14.99), once voted the most incompetent film of all time. The rental sector meanwhile provides — along with all box-office successes which nowadays transfer to tape within few months and probably need no further introduction — a chance to catch up on variety of (often more deserving) movies which have been less widely seen in cinemas here.

Competitive markets are essential if the market system is to operate effectively. In monopoly situation, the producer may be able to use his market power at expense of the consumer, although likelihood of this happening will be moderated if close substitutes for product exist or if the monopolist fears entry of new, but similar, products into market, attracted by high prices and profits being made. It is relatively rare for a firm to have the absolute monopoly of the market, but there are many instances where a small number of large firms dominate the market — this is called oligopoly. Such firms may act overtly (and illegally) or covertly (still illegally, but hard to detect) as if they were in the monopoly situation.

Agreements or understandings could cover price-fixing and/or sharing out available work without resorting to a competition (which would have resulted in lower prices). In order for an agreement to stick, no single firm must break ranks, encouraged by the prospect of a greater market share by lowering its price, or else free-for-all may develop producing a competitive price, as if the market had been operating smoothly. The operation of such a price-fixing agreement can be seen in way that price of oil has been fixed by Organisation of Petroleum Exporting Countries (OPEC) cartel.

High prices were maintained from 1974 onwards in face of the inelastic de-

mand for oil from oil-importing countries, but the oil glut in 1980s put pressure on individual countries to reduce their prices in order to gain a market share. Despite attempts to prevent this by OPEC, the price of oil plummeted in 1986. Even if there are no formal agreements to interfere with the market, implicit understandings may be reached, in which a case behaviour of the firms involved might, in practice, be the same as if a formal agreement had existed.

4.5.4 Annotator C

Sell-through, as the retail sector of the video market is obscurely known, is at the moment very much a bull market, and there has lately been a growing shift to releasing more recent movies in this way. This has been capped by the well publicised current appearance of no less than the Rain Man (Warner) for sale (£14.99) as well as for rent, when hitherto the two spheres have tacitly been regarded as mutually exclusive. But for all the changes, prime emphasis in the retail market, so far as movies are concerned, still seems to be on the past, and predominantly on mainstream Hollywood past.

Current refinements, no doubt with Christmas present buying in mind, include the boxed sets of three or more related cassettes from Warners — for example, all three of James Dean's movies for £29.95, or for the same price, Marilyn Monroe in Prince and the Showgirl , Misfits and (a bit of a scoop , since it has not been on video before) Some Like It Hot. This kind of collection, though usually available individually, is increasingly and intelligently proving to be a marketing ploy. For random instance, CBS/Fox has recently released (£9.99 each) not only Cronenberg's 1986 Fly and its far from negligible 1958 forerunner, but also the 1959 quick-buck sequel to the latter, Return Of Fly, a collector's item, though not necessarily in qualitative terms.

And to move from B to Z movies, connoisseurs of the bizarre can now lay in their own copy of Edward Woods' Plan 9 From Outer Space (Palace, £14.99), once voted the most incompetent film of all time. The rental sector meanwhile provides — along with all box-office successes which nowadays transfer to tape within a few months and probably need no further introduction — chance to catch up on a variety of (often more deserving) movies which have been less widely seen in cinemas here.

Competitive markets are essential if the market system is to operate effectively. In a monopoly situation, the producer may be able to use his market power at the expense of the consumer, although the likelihood of this happening will be moderated if close substitutes for the product exist or if the monopolist fears entry of new, but similar, products into the market, attracted by high prices and

profits being made. It is relatively rare for a firm to have absolute monopoly of the market, but there are many instances where a small number of large firms dominate the market — this is called oligopoly. Such firms may act overtly (and illegally) or covertly (still illegally, but hard to detect) as if they were in a monopoly situation.

Agreements or understandings could cover price-fixing and/or sharing out available work without resorting to competition (which would have resulted in lower prices). In order for an agreement to stick, no single firm must break ranks, encouraged by the prospect of a greater market share by lowering its price, or else a free-for-all may develop producing a competitive price, as if the market had been operating smoothly. Operation of such a price-fixing agreement can be seen in the way that the price of oil has been fixed by the Organisation of Petroleum Exporting Countries (OPEC) cartel.

High prices were maintained from 1974 onwards in the face of an inelastic demand for oil from oil-importing countries, but an oil glut in the 1980s put pressure on individual countries to reduce their prices in order to gain a market share. Despite attempts to prevent this by the OPEC, the price of oil plummeted in 1986. Even if there are no formal agreements to interfere with the market, implicit understandings may be reached, in which case the behaviour of firms involved might, in practice, be the same as if a formal agreement had existed.

4.5.5 Annotator D

Sell-through, as the retail sector of the video market is obscurely known, is at the moment very much a bull market, and there has lately been a growing shift to releasing more recent movies in this way. This has been capped by the well publicised current appearance of no less than Rain Man (Warner) for sale (£14.99) as well as for rent, when hitherto two spheres have tacitly been regarded as mutually exclusive. But for all the changes, a prime emphasis in the retail market, so far as movies are concerned, still seems to be on the past, and predominantly on the mainstream Hollywood past.

The current refinements, no doubt with Christmas present buying in mind, include boxed sets of three or more related cassettes from Warners — for example, all three of James Dean's movies for £29.95, or for the same price, Marilyn Monroe in Prince and Showgirl , the Misfits and (a bit of a scoop , since it has not been on video before) Some Like It Hot. This kind of collection, though usually available individually, is increasingly and intelligently proving to be a marketing ploy. For random instance, CBS/Fox has recently released (£9.99 each) not only Cronenberg's 1986 the Fly and its far from negligible 1958 forerunner, but also

the 1959 quick-buck sequel to the latter, Return Of Fly, a collector's item, though not necessarily in qualitative terms.

And to move from B to Z movies, connoisseurs of the bizarre can now lay in their own copy of Edward Woods' Plan 9 From Outer Space (Palace, £14.99), once voted the most incompetent film of all time. The rental sector meanwhile provides — along with all box-office successes which nowadays transfer to tape within a few months and probably need no further introduction — a chance to catch up on a variety of (often more deserving) movies which have been less widely seen in cinemas here.

Competitive markets are essential if a market system is to operate effectively. In a monopoly situation, a producer may be able to use his market power at the expense of the consumer, although the likelihood of this happening will be moderated if close substitutes for the product exist or if the monopolist fears entry of new, but similar, products into the market, attracted by high prices and the profits being made. It is relatively rare for a firm to have an absolute monopoly of a market, but there are many instances where a small number of large firms dominate a market — this is called an oligopoly. Such firms may act overtly (and illegally) or covertly (still illegally, but hard to detect) as if they were in a monopoly situation.

Agreements or understandings could cover price-fixing and/or sharing out available work without resorting to competition (which would have resulted in lower prices). In order for an agreement to stick, no single firm must break the ranks, encouraged by the prospect of greater market share by lowering its price, or else a free-for-all may develop producing a competitive price, as if the market had been operating smoothly. The operation of such a price-fixing agreement can be seen in the way that the price of oil has been fixed by the Organisation of Petroleum Exporting Countries (OPEC) cartel.

High prices were maintained from 1974 onwards in the face of inelastic demand for oil from oil-importing countries, but the oil glut in the 1980s put pressure on individual countries to reduce their prices in order to gain a market share. Despite attempts to prevent this by OPEC, the price of oil plummeted in 1986. Even if there are no formal agreements to interfere with the market, implicit understandings may be reached, in which case the behaviour of firms involved might, in practice, be the same as if a formal agreement had existed.