Ellipsis Detection: Machine Learning vs. Rule-Based Document Classification

Master's Thesis

by

Miriam Kümmel

Speech and Language Processing

A Thesis Presented for the Master of Arts Degree

1. Supervisor: Prof. Dr. Miriam Butt

2. Supervisor: Prof. Dr. George Walkden

Konstanz, 2019

ABSTRACT

Sluices are a recurrent form of ellipsis where all but the interrogative pronoun of an embedded question gets elided, leaving the wh-pronoun as the final word of the sentence. However, especially in German, not all sentences with a final wh-word are sluices.

The goal of this thesis was the development of a rule-based algorithm that successfully and fully automatically distinguishes between sluices and non-sluices in German wh-final sentences. The rules of the algorithm are able to generalize a sentence into one category only by means of surface-syntactical cues and the use of part-of-speech-tags.

In the course of algorithm development, a richly annotated and part-of-speech-tagged corpus of 479 wh-final sentences was created. It evolved into a valuable resource for the (prospective) exploration of wh-final sentences and sluices.

The performance of the algorithm was compared to the performance of a machine learning (ML) document classification algorithm by Susan Li (2018)[1].

While the rule-based algorithm turned out to deliver significantly more accurate classification results, the ML algorithm proved to be superior in other respects such as resource efficiency and implementation effort.

[1] https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f

ACKNOWLEDGEMENTS

This research was supported by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD).

I would like to thank my advisor Prof. Rajesh Bhatt for accompanying me through the entire creation process of the thesis. His advice, contributions, and patience in answering my questions were always greatly appreciated. I am also grateful for discussions with Jennifer Spenader, Simone Teufel and Mohit Iyyer.

For enabling this project and the support in all respects I would like to thank my professor Prof. Dr. Miriam Butt.

1. INTRODUCTION

1.1 Overview: Ellipses and their Computational Processing

Different forms of ellipses – sentences with missing subparts (Anand and Hardt 2016: 1234) – are pervasive in natural language. Redundant, thus unnecessary material can often be omitted, resulting in an expression of meaning without form. The use of ellipses, 'is […] an obvious and natural method to exploit redundancies in a system while maintaining usability' (Merchant 2001: 2).

The object of omission can be predicate material (1), nominal material (2) or sentential material (3): Some of the best investigated instances of ellipsis are verb phrase ellipsis (1) (VPE, henceforth), nominal ellipsis (2) and sluicing (3) (van Craenenbroeck and Merchant 2013: 701–731)[2].

(1) John likes candy, but Bill doesn't [like candy].

(2) Früher trank ich deutsches Bier, aber heute nur noch spanisches [Bier].
Earlier drank I German Beer, but today only Spanish [beer].

(3) Ed tötete jemanden, aber ich weiß nicht, wen [Ed tötete].
Ed killed someone, but I know not who [Ed killed].

Natural 'languages differ extensively in how they allow redundancies to be reduced by the grammar' (Merchant 2001: 2): While VPE occurs in English but not in German[3], and nominal ellipsis occurs frequently in German but rarely in English (GECCo Project 2017: 6), sluicing is common in both languages and many more (Merchant 2003: 1).

Even though ellipses are 'more work' for a hearer, as their 'interpretation is richer than what is actually pronounced', employing and interpreting ellipses usually poses no problems for human language users and even increases the efficiency of their communication (Merchant 2001: 1; Aelbrecht 2009: 1). However, it introduces challenges for machine language processing technologies as certain aspects of text meaning are not traceable to surface text elements (McShane and Babkin 2015: 1). Where meaning is bound to seeming emptiness, machines struggle to perform. For a functioning computational processing of language input, ellipses must be resolved, i.e. the missing material must be rebuilt.

[2] All examples taken from literature (and sometimes slightly modified) are referenced. If there's no reference, the example is either a translation of a referenced example or taken from one of the created corpora (and as well, sometimes modified).
[3] In fact, VPE are 'quite rare among the world's languages' Merchant (2001: 3).

An example for the struggle of machines dealing with sluices can be seen in the parsing output of (3) (Figure 2) and its rebuilt version (Figure 1) by the XLE parser (Rosén *et al.* 2012):

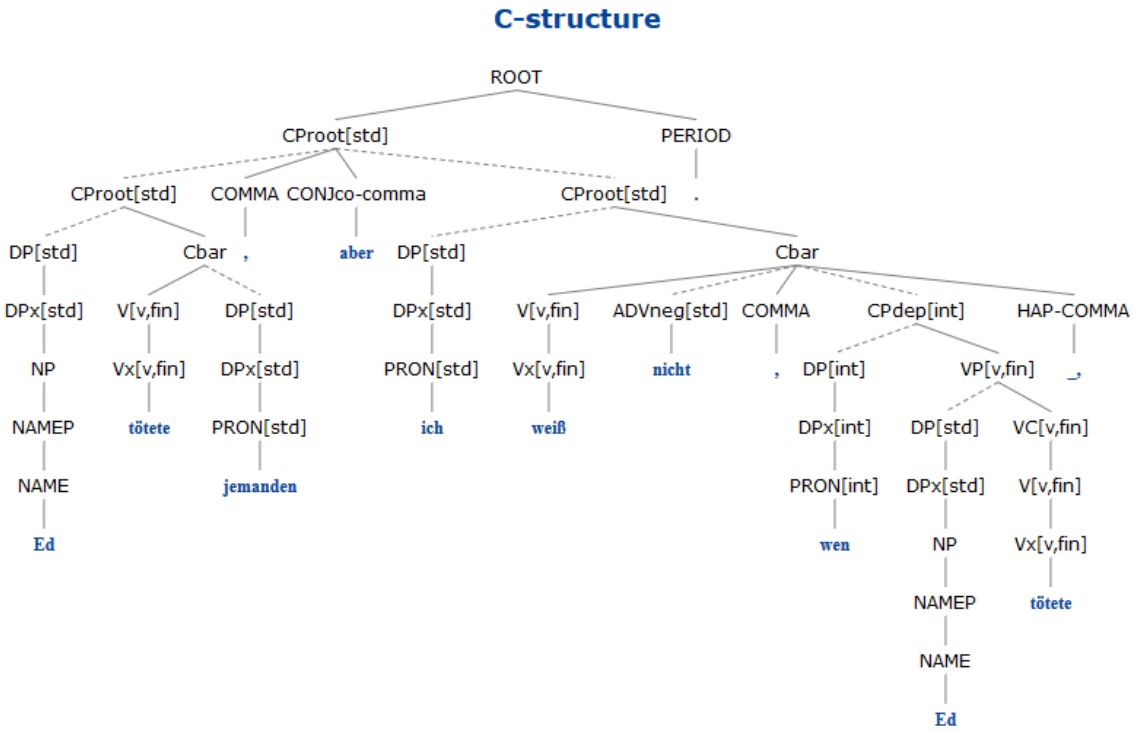


Figure 1: Parsing output of non-elliptical sentence

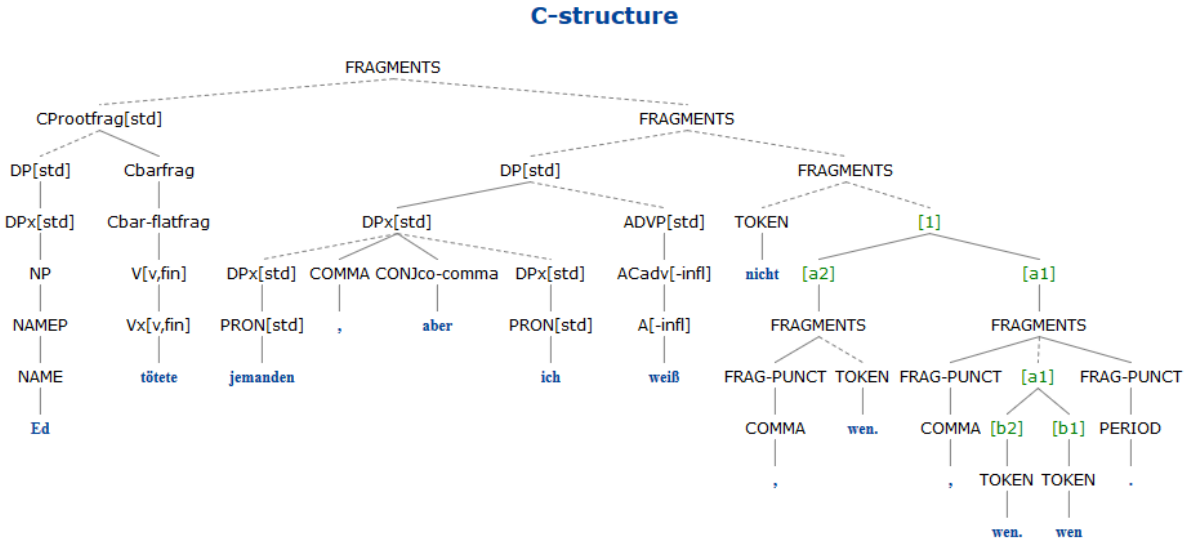


Figure 2: Parsing output of sluice: Ed tötete jemanden, aber ich weiß nicht, wen [ ].

The non-elliptical sentence in Figure 1 led to a correct parse; all phrases were assigned the correct structure, 'nicht' modifies 'weiß', 'wen Ed tötete' is dependent on 'ich weiß nicht,' and so on.

The sluice in Figure 2, on the other hand, could not be processed correctly. Subject ('Ed') and object ('jemanden') of the sentence were assigned to different CP, 'weiß' was labelled as an adverb phrase[4], 'jemanden, aber ich weiß' was recognized as a DP, etc. The ellipsis at the end of the sentence makes the parse of the whole sentence collapse spectacularly; correct parsing is not possible if there is a sluice in the sentence. Looking back at the excellent parse in Figure 1, it is clear that the gap in a sluice needs to be rebuilt in order to process sentences like these correctly.

Many, if not all, natural language processing (NLP, henceforth) applications, such as automatic document summarization or question answering, rely on the ability to process and understand[5] natural language input produced by humans. Since in human communication, the usage of ellipses is omnipresent, resolving ellipses is crucial for NLP systems to run smoothly (Nielsen 2005: 28).

An ellipsis resolution algorithm involves three successive subtasks (Nielsen 2005: 28):

1. Detecting ellipsis occurrences
2. Identifying antecedents
3. Resolving the ellipsis

The performance of each step is highly dependent on the performance of the previous steps – errors made in the first two steps will accumulate on their way up. This thesis will focus on the task of detecting sluices.

Sluices are a cross-linguistically common form of ellipsis (Merchant 2003: 1): they are an embedded question where all material except for the **wh-word** has been elided.

(4)	Sie	beleidigte	jemanden,	aber	sagt	mir	nicht,	**wen**	[sie	beleidigte]**.**
	She	*insulted*	*somebody,*	*but*	*tells*	*me*	*not*	***who***	*[she*	*insulted].*

Here, the VP 'sie beleidigte' 'is not phonetically realized because its meaning is recoverable from the [sponsor clause][6]' (Aelbrecht 2009: 1), leaving the interrogative pronoun 'wen' stranded at the end of the sentence.

A lot of the sentences ending in the pattern WH+PERIOD are sluices (see (4)). But there are also many sentences with this pattern that are not sluices, but e.g. sentence-final indefinite pronouns. Here, no material is missing:

(5)	Herr	Lehmann,	ich	sag	dir	mal	was.
	Mister	*Lehmann,*	*I*	*tell*	*you*		*something.*

[4] 'weiß' is ambiguous in German (*engl.: white* and *know*$_{3.PersSg}$)
[5] Natural Language Understanding (NLU) is considered a large subfield of NLP.
[6] See 2.2 for terminology.

A computer cannot make the distinction between sluices and non-sluices unaided.

The first step in the investigation of a syntactic phenomenon is building a collection of sentences which displays it. In this thesis, a corpus containing 479 sentences with a final wh-word was built and thoroughly annotated. Among other things, it proved that sluices cannot be detected in a text collection by just looking for sentence-final question words, but more sophisticated distinctions must be made.

The focus of the present thesis is the development of an algorithm that can reliably distinguish between sluice and non-sluice in German wh-final sentences. This way, it can facilitate, for example, the automatic creation of a corpus of sluices for future investigations of the phenomenon or serve as a first step in sluice resolution. In order to be able to assess this algorithm's performance, a supervised machine learning document classification algorithm (Li 2018) was executed on the data as well and the two algorithms' characteristics and performance were compared.

1.2 Outputs and Algorithmic Structure of the Thesis

This thesis will produce two outputs:

- A partly hand-annotated, partly computationally annotated and POS-tagged (TreeTagger (Schmid 2018)) corpus of 479 German sentences with a final wh-word (`whfinals_corpus.csv`)
- A rule-based algorithm that classifies wh-final sentences as sluice or non-sluice, relying on only automatically generated meta-data (`rule_based_doc_class.py`), together with a comparison to a machine learning document classification algorithm (`ML_doc_class.py`)

The following flowchart depicts the processing and reshaping of the initial data and the relation of the different algorithms:

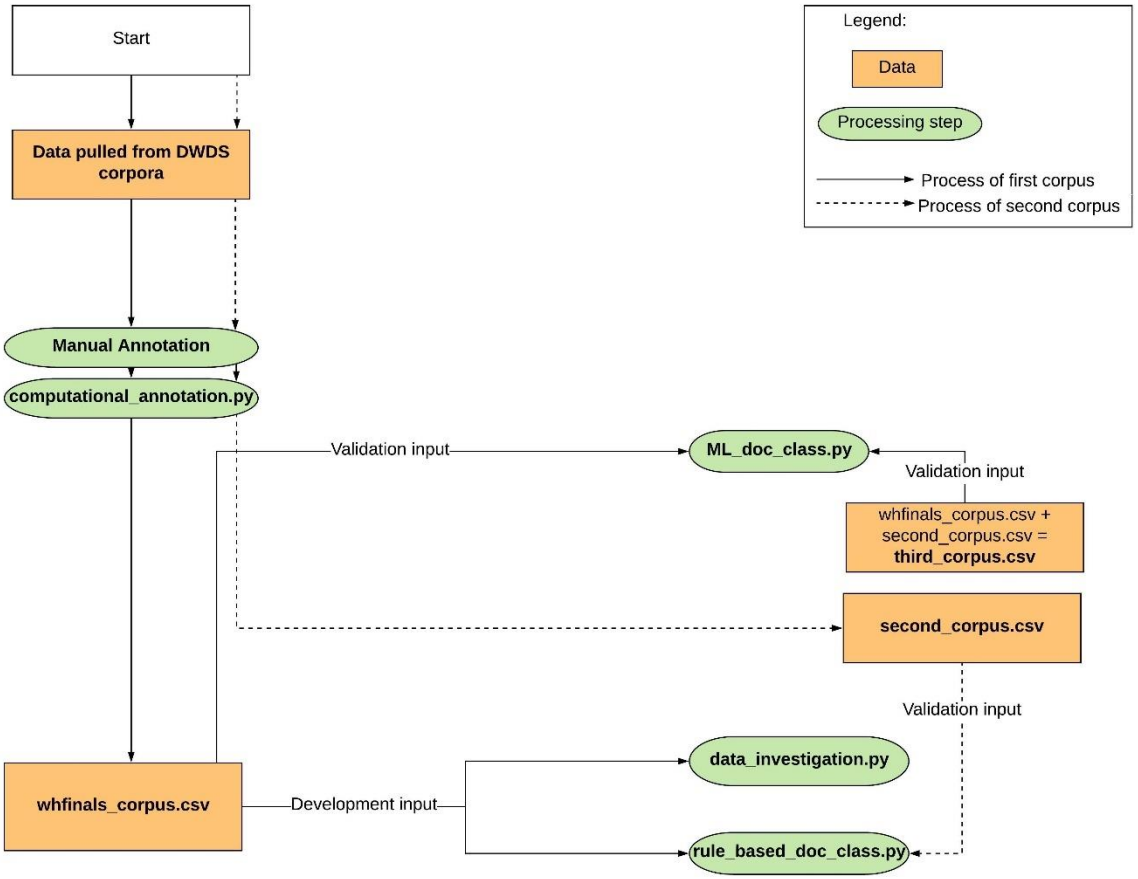


Figure 3: Algorithmic structure of the thesis

After pulling wh-final sentences from the DWDS corpora (Klein 2018) (see chapter 4), the sentences were manually annotated with several information, inter alia, the tag sluice/non-sluice (see 5.1.1). Afterwards, the sentences and the respective previous sentence were POS-tagged (see 5.2.1). With the help of the POS-tags, the computational annotation could be executed (see 5.2.2 and following) which facilitated the investigation of the data (see chapter 5) and, in particular, the development of the rule-based document classification algorithm (see 6.3).

The rule-based algorithm was subsequently tested, and its performance validated with the second corpus (see 4.4.1), which provided data unfamiliar to the developer; the ML algorithm was tested and validated with the wh-finals corpus itself, and, additionally with the third corpus which combined the wh-finals corpus and the second corpus (see 4.4.2).

2. THEORY: SLUICING

To be able to develop sensible rules for sluice detection, the theory behind the phenomenon must be clear. After a general introduction to the term (2.1-2.3) an overview over different forms of sluices and their characteristics will be given (2.4-2.5).

2.1 The Phenomenon of Sluicing

Sluicing is a sentential or clausal[7] ellipsis phenomenon (van Craenenbroeck and Merchant 2013: 718)[8]. Just like in other elliptical phenomena the usual form-meaning correspondence of an utterance seems to break down: meaning is attached to blankness.

Ross (1969) first described and named this 'operation of a rule' as the following transition (Ross 1969: 252; Merchant 2003: 1).

(6a) He is writing but you can't imagine what he is writing.
(6b) He is writing but you can't imagine what [ ].

The rule of sluicing applied on (6a) results in a deletion of all sentential material of the embedded question '…what he is writing' except for its preposed constituent 'what' (Ross 1969: 252) leading to a stranded interrogative phrase instead of a complete constituent question (Chung 2005: 73). The meaning of (6a) and (6b) is identical, the sluiced wh-phrase in (6b) conveys a full interrogative force (Algryani 2011-2012: 42). This rule can only operate on (embedded) constituent questions, not on whether-clauses (Ross 1969: 272).

While there is an ongoing debate about whether the elided material in a sluice can be assigned syntactic structure or not (Merchant 2003: 2), the 'opinio communis' (Merchant 2001: 53) on this topic is the following: sluicing is the ellipsis of IP (inflectional phrase) in a constituent question. What surfaces as a stranded wh-phrase is therefore a full interrogative CP (complementizer phrase) (Barros 2014: 1; Merchant 2001: 53) that underwent a deletion process even though the deleted material does not form a constituent (Ross 1969: 253) (see Figure 4).

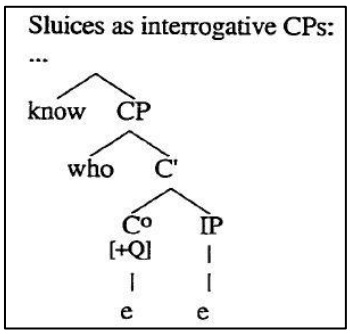


Figure 4: The syntax of sluicing (Merchant 2001: 54)

[7] As opposed to predicate ellipsis phenomena like VPE (van Craenenbroeck and Merchant (2013: 702)) and nominal ellipsis (van Craenenbroeck and Merchant (2013: 732)).
[8] Other clausal ellipsis phenomena are stripping (Ed likes stiletto heels and Maggy too.), gapping (John likes sandals and Mary stiletto heels.), fragment answers (A: What did you buy? B: A boat.) and null complement anaphora (Ed wanted Bill to help Mary, but he refused.) (van Craenenbroeck and Merchant (2013: 718)).

2.2 Terminology

The stranded interrogative phrase will be referred to as remnant, the elided part of the constituent question is the (elided) IP (Chung 2005: 73), the whole embedded question is a CP. The clause that provides the material that was elided will be called sponsor (clause) (McShane and Babkin 2016: 1), the verb licensing the sluice is the **embedding verb** which is part of the embedding clause. If there's an overt phrase the elided IP correlates to (this is not mandatory, see 2.4.3), it will be called antecedent (Merchant 2003: 1). The antecedent and the sponsor clause produce the Presluice:

(7a)	Jack	kaufte	etwas,	aber	ich	weiß	nicht,	was	Jack	kaufte.
	Jack	*bought*	*something*	*but*	*I*	*know*	*not*	*what*	*he*	*bought.*
(7b)	Jack	kaufte	etwas,	aber	ich	**weiß**	nicht,	was	[ ].	
	I	*eat*	*something*	*but*	*I*	*know*	*not*	*what.*		

2.3 Sluicing in German

Unlike other greatly researched ellipsis phenomena, like VPE, sluicing is linguistically widespread and can be found in a lot of different languages (Merchant 2003: 1). From a surface-syntactical viewpoint, sluicing in English and German share most of their characteristics, as it can be seen in this direct translation of (6a) and (6b):

(8a)	Er	schreibt,	aber	du	kannst	dir	nicht	vorstellen,	was	er schreibt.
(8b)	Er	schreibt,	aber	du	kannst	dir	nicht	vorstellen,	was	[ ].

Apart from the comma between the embedding clause and the CP, which is orthographically mandatory but was found to be omitted in a lot of the cases in the corpora developed in this thesis, the structure is identical.

A difference between English and German grammar that plays a role in sluicing is case marking: In German, the wh-remnant of a sluice must bear the same case that the antecedent bears (Merchant 2001: 183)[9], if it can bear a case at all and if an antecedent exists at all: in (7b), for example, 'etwas' and 'was' are both in the accusative.

Two of all wh-phrases (the substituting interrogative pronouns (Schmid 2018)) can bear case: 'wer' and 'welche' (*Engl.: who and which*), all other wh-phrases are not inflectional.

Nominative	**wer/was**	**welche(r/s)**
Genitive	wessen	welche(r/s)
Dative	wem	welchem(-r/-n)
Accusative	wen/was	welche(n/s)

Table 1: Case bearing wh-words in German

[9] This is not the case for every case bearing language (see Vicente (2015)).

2.4 Typology of Sluices

2.4.1 The Simplest Sluice

The syntactically most simple form of sluicing can be seen in (9)[10].

(9)	Jack	rannte,	aber	ich	weiß	nicht,	warum	[Jack	rannte].
	Jack	*ran*	*but*	*I*	*know*	*not*	*why*	*[Jack*	*ran].*

The verb in the sponsor clause is intransitive (therefore there are no objects). There is no overt antecedent and no additional material like PPs or other adjuncts. The elided IP and the sponsor clause are the exact same material in the same order ('Jack rannte') and the CP solely consists of the wh-phrase and the sponsor clause. The elided IP can easily be rebuilt by inserting the sponsor of the sentence.

2.4.2 Sentence-Final vs. Sentence-Internal Sluices

There are several reasons why the wh-remnant of a sluice could **not** be followed by punctuation or an adjoining conjunction and therefore form a sentence-internal sluice.

2.4.2.1 NP Stays in Place

The nominal phrase (NP) of the elided IP can stay in place (Ross 1969: 255), resulting in a WH+NP-pattern:

(10)	Er	gibt	uns	ein	Problem,	aber	es	ist	nicht	klar,	welches	Problem.
	He	*gives*	*us*	*one*	*problem,*	*but*	*it*	*is*	*not*	*clear*	*which*	*problem.*

Very little proof or description can be found in the literature for instances of such seemingly half-resolved sluice: while the subject and predicate of the sponsor clause are still missing, the accusative object 'Problem', which also functions as antecedent of the sluice, has been rebuilt into the gap. This construction is only licensed by the wh-phrases 'welche' and 'wessen' (*Engl.: which* and *whose*) as they are the only wh-phrases that can build an DP together with a noun. Sentence (10) technically contradicts Ross' (1969) rule of sluicing, which is defined by: '… deleting everything but the preposed constituent of an embedded question' (Ross 1969: 252). Later authors, too, emphasized the significance of the lone wh-remnant: '[…] leaving only a wh-remnant' (Merchant 2003: 1), '[…] interrogative in which all but the questioned element is missing' (Dayal and Schwarzschild 2010: 92), '[…] occurs when the target clause contains only an embedded WH- phrase and an inflectional phrase (a verb phrase with a finite form)' (Bos and Spenader 2011: 469), '[…] all but the interrogative phrase of a content question is elided' (Anand and McCloskey 2015a: 1). Still, Ross (1969) himself instances this example without

[10] Examples (7a-b) and (9) are made up.

further elaborating on the difference to all his other examples (Ross 1969: 255) which is the reason to consider it a sluice, too.

2.4.2.2 Constituent Reordering

A reorganisation of constituents can also lead to sentence-internal sluices (Anand and McCloskey 2015a: 1). Here, the ellipsis can be found in the middle of the sentence:

(11a)	Es	ist	klar,	dass	sich	die	Uni	ändern	muss,	aber	wie	[ ]	ist	weniger	klar.
	It	*is*	*clear,*	*that*	*itself*	*the*	*uni*	*change*	*must,*	*but*	*how*	*[ ]*	*is*	*less*	*clear.*

The deletion process is the same as in sentence-final sluices: the elided material ('sich die Uni ändern muss') is an IP, the wh-remnant 'wie' stays in place. What changed in this case is the phrase order of the sentence: The embedding clause and its sluice-licensing verb 'be clear' is now found after the elided IP. The process can be reversed (see (11b)). However, not every sentence can be transferred from one shape into the other just by shifting around its phrases. For example, converting (11a) into a sentence-final sluice requires an additional DP ('es') in the embedding clause (Anand and McCloskey 2015a: 1).

(11b)	Es	ist	klar,	dass	sich	die	Uni	ändern	muss,	aber	**es**	ist	weniger	klar,	wie	[ ].
	It	*is*	*clear,*	*that*	*itself*	*the*	*uni*	*change*	*must,*	*but*	*it*	*is*	*less*	*clear*	*how*	*[ ].*

Turning (9) into a sentence-internal sluice, on the other hand, results in the inversion of the subject ('ich') and the predicate ('weiß') of the embedding clause:

(9b)	Jack	rannte,	aber	warum	[Jack	rannte],	weiß	ich	nicht.
	Jack	*ran*	*but*	*why*	*[Jack*	*ran],*	*know*	*I*	*not.*

2.4.2.3 Not a Reason: Swiping

Swiping (an acronym for **s**luiced **w**h-word **i**nversion with **p**repositions **i**n **n**orthern **G**ermanic (Merchant 2003: 15)) is the inversion of a preposition and the wh-remnant that is governed by it. It can only be found in sluicing, not in any other ellipsis phenomenon (Merchant 2003: 16). While in English and some other Germanic languages (Anand and McCloskey 2014: 11), swiping can be another reason for a wh-remnant not to be the last element of a sentence, it does not occur in German (Anand and McCloskey 2014: 11).

(12)	He	went	to	the	movies,	but	I	don't	know	who	with.
	*Er	ging	ins		Kino,	aber	ich	weiß	nicht,	wem	mit.

2.4.3 Correlates: Merger vs. Sprouting

Sluices can further be divided into two subtypes of referencing: Merger and Sprouting. In Merger sluices, the wh-remnant refers to an overt correlate (= antecedent) in the main clause (Chung, Ladusaw and McCloskey 2011: 1) that, with few exceptions (Dayal and Schwarzschild

2010: 112), is a semantically indefinite phrase or expression (Anand and Hardt 2016: 1235) like 'jemanden' (*Engl.*: *somebody*):

(13)	Sie	beleidigte	**jemanden**,	aber	sagt	mir	nicht,	wen	[sie	beleidigte].
	She	*insulted*	***somebody***,	*but*	*tells*	*me*	*not*	*who*	*[she*	*insulted].*

In Sprouting, there is no antecedent the wh-remnant correlates to:

(14)	Sie	lächelte	und	wusste	nicht,	warum	[sie	lächelte].
	She	*smiled,*	*and*	*knew*	*not*	*why*	*[she*	*smiled].*

This distinction could be seen in correlation to the transitivity of the respective verb in the main clause. (Di-)transitive verbs like 'beleidigen' in (13) cannot occur without an object (like 'jemanden'). But even a (di-)transitive verb can build a Sprouting sluice: namely, if the wh-remnant does not correlate with the object ('jemanden'). In this case, the object is not the antecedent of the sluice and therefore the sentence is an instance of Sprouting.

(13b)	Sie	beleidigte	jemanden,	aber	sagt	mir	nicht,	**warum**	[sie	jemanden	beleidigte].
	She	*insulted*	*somebody,*	*but*	*tells*	*me*	*not*	*why*	*[she*	*somebody*	*insulted].*

Furthermore, if the wh-remnant ('warum') does not correlate to the object ('jemanden'), the object becomes a part of the elided IP which is not the case in Merger (see (13)). This leads to the following rules:

I	transitive verb	+	object	+	correlating wh-phrase	→ no object in elided IP	**(= Merger)**
II	intransitive verb					→ no object in elided IP	**(= Sprouting)**
III	transitive verb	+	object	+	not-correlating wh-phrase	→ object in elided IP	**(= Sprouting)**

Table 2: Transitivity of Verbs - Merger or Sprouting?

2.4.4 Multiple Sluicing

A sentence can show more than one wh-remnant (Barros 2014: 38). The two remnants often occur in direct sequence:

(15a)	Die	Liebe	ist	verlorengegangen	und	ich	weiß	nicht,	wo	[ ]	und	wann	[ ].
	The	*love*		*went missing*	*and*	*I*	*know*	*not*	*where*	*[ ]*	*and*	*when*	*[ ].*

The elided IPs of both remnants are identical:

(15b)	[…]	wo	die Liebe verlorengegangen ist	und	wann	die Liebe verlorengegangen ist.
	[…]	*where*	*the love went missing*	*and*	*when*	*the love went missing.*

2.4.5 Full CP with Sluice

Oftentimes a sluiced wh-remnant follows a full CP. Here, the structure is similar to the structure in multiple sluicing (see 2.4.4), the sluice questions the same phrase as the full CP.

(16) Keiner weiß, warum sie gewandert sind und wann [sie gewandert sind].
Nobody knows why they wandered and when [they wandered].

If the first - unelided - CP questions a mandatory phrase of the sponsor (the subject or an object), the rebuilt IP includes the wh-phrase of the full CP:

(17) Sie erinnern sich nicht, was sie entschieden haben und warum
They remember not what they decided have and why

[sie **was** entschieden haben].
*[they **what** decided have].*

2.4.6 Embedded vs. Unembedded Sluices

While all the previous examples and the vast majority of the corpus data are embedded sluices, embedding is no requirement for sluicing. Unembedded sluices ('root sluices' (Anand and McCloskey 2015a: 3)) do not incorporate a licensing verb, i.e. an embedding verb. Especially easy to spot are the ones that only contain an isolated wh-remnant and happen between two speakers (Anand and McCloskey 2014: 14):

(18) A: Wir sollten nach Hause gehen. B: Warum [sollten wir nach Hause gehen]?
A: We should home go. B: Why [should we go home]?

But they can be part of a larger phrasal context, too:

(19) Sie forderte mich auf, aber wozu [forderte sie mich auf].
She urged me but to what. [did she urge me].

Unembedded sluices are questions, not declarative sentences which is, of course, reflected in the constituent order of the elided IP.

2.4.7 PP-Adjuncts in Antecedents and Prepositions in Remnants

A lot of verbs allow PP-attachment. At the same time, not all wh-phrases are able to take a preceding preposition. Regarding PP-attachment, the wh-phrases split up in three groups:

Group	Wh-Phrase	Combination with Preposition	Example[11]
I	was, welche/-r/-m/-n, wessen, wem, wen, wie viele, wann, wo	some prepositions possible	Er kommt, aber mit wem? *He comes, but with who?*
II	warum/wieso/weshalb/weswegen, wer, wie, wieweit, welche[12]	no preposition possible	*Er kommt, aber ANY_PREP warum? **He comes, but ANY_PREP why?*

[11] For reasons of space, examples are made up unembedded sluices.
[12] 'welche' and 'wer' in their nominative forms, cannot occur with a preposition, as a preposition always entails a case.

III	wobei, wodurch, wofür, …	preposition is incorporated	Er kommt, aber wofür? *He comes, but what for?*

Note that for case government reasons and semantic reasons, the wh-phrases in group (I) cannot be combined with just any preposition ('*Er kommt, aber seit wem?' (*Engl.: '*He comes, but since who?*'), but only specific ones (Zwarts 2005: 12).

If the sponsor comes with a prepositional adjunct ('mit jemandem' in (14)), this adjunct can potentially serve as an antecedent if the same preposition is found in the remnant (Merger, see (20a)). If there's no preposition in the remnant (see (20b)), the remnant cannot be correlated to the antecedent (Sprouting) (Merchant 2012: 18). But even if there is a preposition in the remnant (see 'seit' in (20c)), it does not have to be the same as in the antecedent and therefore does not have to be correlating to the antecedent. (20c) thus, is an instance of Sprouting, too (van Craenenbroeck and Merchant 2013: 726).

(20a)	Er	hat	mit	jemandem	gesprochen,	aber	ich	weiß	nicht,	mit	wem.	(Merger)
	He.	*has*	*with*	*someone*	*spoken*	*but*	*I*	*know*	*not,*	*with*	*who.*	
(20b)	Er	hat	mit	jemandem	gesprochen,	aber	ich	weiß	nicht,	warum.		(Sprouting)
	He	*has*	*with*	*someone*	*spoken*	*but*	*I*	*know*	*not,*	*why.*		
(20c)	Er	hat	mit	jemandem	gesprochen,	aber	ich	weiß	nicht,	seit	wann.	(Sprouting)
	Er.	*has*	*with*	*someone*	*spoken*	*but*	*I*	*know*	*not,*	*since*	*when.*	

Verbs in the presluice can occur with or without their PP-adjunct which sometimes leads to an absent PP-adjunct in the presluice and a present preposition in the remnant. As there's no overt antecedent then, (17) is an example of Sprouting.

(21)	Das	ist	wichtig,	sie	begreift	nur	nicht,	für	wen.	(Sprouting)
	That	*is*	*important,*	*she*	*understand*	*only*	*not*	*for*	*who*$_{ACC.}$	

A 'PP-free' presluice therefore does not automatically result in a preposition-free remnant.

2.5 Licensing of Sluices

While VPE ('John likes candy, but Bill doesn't [ ].'(van Craenenbroeck and Merchant 2013: 702)) can only be licensed by some auxiliaries (Anand and Hardt 2016: 1235), the group of sluice-licensing verbs is a lot more difficult to specify. As sluices are, in fact, indirect (as opposed to direct questions) wh-questions (as opposed to alternative questions) (Karttunen 1977: 4), they can only be licensed by CP-licensing verbs: 'All and only predicates that s-select questions and c-select CPs allow sluiced wh-phrases'[13] (Merchant 2001: 54).

In the perspective of licensing embedded questions, verbs can be split in three categories (Roelofsen, Theiler and Aloni 2017: 4):

[13] For more information about c- and s-selection, see Fromkin *et al.* (2000: 197).

Type of verb	Can take…	Examples	
Responsives	declarative + interrogative complements	declarative	Mary knows that John left.
		interrogative	Mary knows who [left].
Rogatives	interrogative complements	declarative	*Bill wonders that Mary left.
		interrogative	Bill wonders who [left].
Anti-Rogatives	declarative complements	declarative	Bill believes that Mary left.
		interrogative	*Bill believes who [left].

Table 3: Types of embedding verbs

Sluices, just like other interrogative complements, must be embedded by either a responsive or a rogative verb like 'know', 'guess', 'remember' etc. (Algryani 2011-2012: 43). The category combining rogative and responsive verbs shall be called question embedding verbs (QEV, henceforth). Karttunen (1977) listed and divided those verbs into semantic categories[14]:

(a) verbs of retaining knowledge: *know, be aware, recall, remember, forget*
(b) verbs of acquiring knowledge: *learn, notice, find out, discover*
(c) verbs of communication: *tell, show, indicate, inform, disclose*
(d) decision verbs: *decide, determine, specify, agree on, control*
(e) verbs of conjecture: *guess, predict, bet on, estimate*
(f) opinion verbs: *be certain about, have an idea about, be convinced about*
(g) inquisitive verbs: *ask, wonder, investigate, be interested in*
(h) verbs of relevance: *matter, be relevant, be important, care, be significant*
(i) verbs of dependancy: *depend on, be related to, have an influence on, be a function of, make a difference to*

Figure 5: Question embedding verbs (Karttunen 1977: 4)

Karttunen (1977) and George (2011), amongst others, elaborated on the semantical classification and characteristics of QEV. Anand and Hardt (2016) describe them to be expressing uncertainty or vagueness about an issue (Anand and Hardt 2016: 1235)[15].

However, there are no shallow cues that facilitate the distinction between anti-rogative verbs and QEV. No morphological or syntactical rule has been stated about which verbs can be QEV and which cannot. Looking at Karttunen's (1977) list and the embedding verbs in the corpus, however, some rules can be derived:

[14] This is not an exhaustive list.
[15] The finding of e.g. 'sagen', 'wissen' and 'entscheiden' (*engl.: say, know, decide*) as unnegated embedding verbs in the corpus contradicts this statement strongly.

1. QEV are **full verbs** (e.g. wissen, *Engl.: to know*) or **functional verbs** (e.g. keine Vorstellung haben, *Engl.: to have no idea*). Copula verbs (e.g. sein, *Engl.: to be*)[16], modal verbs (e.g. können, *Engl.: to can*), modality verbs (e.g. scheinen, *Engl.: to seem*) or auxiliary verbs cannot be QEV.
2. QEV are either **transitive or ditransitive** verbs. Intransitive (e.g. rennen, *Engl.: to run*) cannot license a question.
3. QEV are **personal verbs**. Impersonal verbs (e.g. regnen, *Engl.: to rain*) cannot license a question.
4. QEV can be both **reflexive** (e.g. sich erinnern, *Engl.: to remember*) or **non-reflexive**.

[16] Note that a copula can serve as a full verb, too. It can still not license a question/sluice.

3. Related Work

Surely one of the most impactful projects on sluices, and an inspiration for the creation of the wh-finals corpus, is the **Santa Cruz Ellipsis Project (Anand and McCloskey 2015b)**. In this project, Anand and McCloskey developed 'a linguistically-informed annotation scheme for sluicing' (Anand and McCloskey 2015a: 1). One of their main goals was to find a 'theory-neutral representation of elliptical content' that can sort out the 'syntactic and semantic mismatches between antecedents and elliptical content'. The authors used a semi-automated sluice detection approach: they parsed a subset of the Gigaword Corpus and then extracted all verb phrases whose final child was a wh-phrase. This yielded 5100 verb phrases, which were manually culled to 4100. Out of those, they annotated 417 instances with four obligatory and additional optional tags:

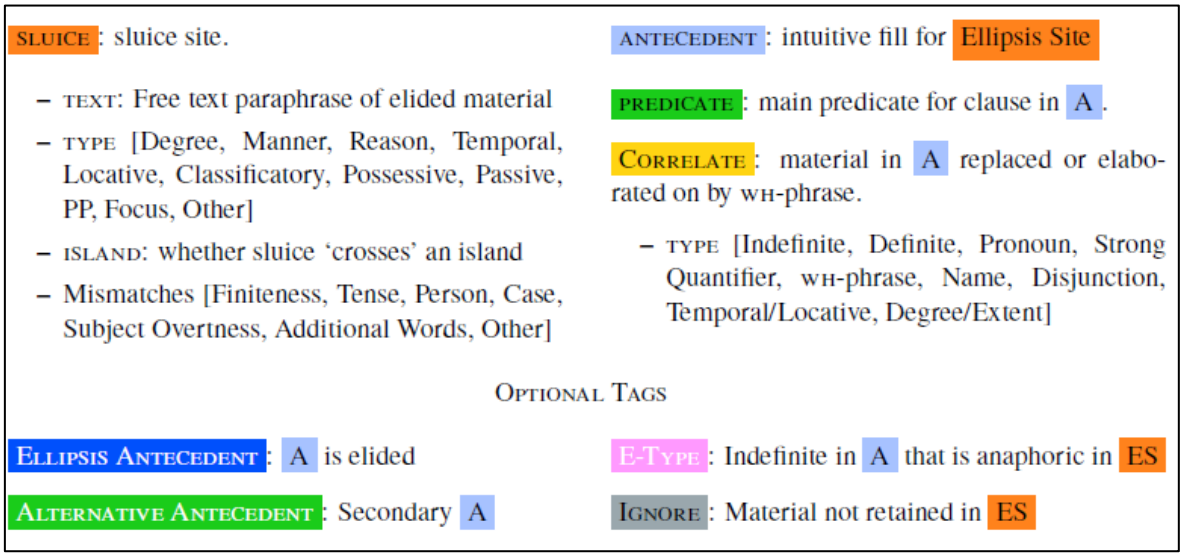


Figure 6: Annotation tagset of Anand and McCloskey 2015a: 5

Their central research question was: What, if anything, is the content of the ellipsis site? And: What type of annotation can facilitate sluice resolution (Anand and McCloskey 2015a: 3)? In general, the goal and outcome of this project was an exhaustively annotated corpus of English sluices.

A methodological inspiration for the thesis, on the other hand, were **McShane and Babkin (2016)** and their attempt to build a VPE detecting and resolving system (**ViPER**) that relies on 'linguistic principles' (McShane and Babkin 2016: 1). Their algorithm involved an ellipsis candidate detector, different ellipsis resolution strategies (rules) and an approach for selecting the sponsor clause of the VPE. Especially their VPE detector posed an interesting role model for the development of the present algorithm: In a first step, ViPER searches a corpus for auxiliary verbs directly followed by a punctuation mark or conjunction. The authors admit that this method cannot offer full recall since VPE are not necessarily followed by a punctuation mark

or conjunction but it did cover many cases and was considered sufficient (McShane and Babkin 2016: 5): ViPER can 'treat at least some instances of VP ellipsis, which is better than treating none' (McShane and Babkin 2016: 25). After finding those candidates, false positives were eliminated by different rules that rely on a parser (McShane and Babkin 2016: 6).

Nielsen (2005) states that the success of an ellipsis resolution algorithm depends on a sensible ellipsis detection step (Nielsen 2005: 28). He also describes how the research work on ellipsis mostly focusses on ellipsis interpretation, while ellipsis detection is usually taken as given (Nielsen 2005: 28). In his doctoral thesis, he claims that 'it is possible to produce the components of a VPE resolution system [VPE detection, antecedent identification, and VPE resolution], using a corpus-based, knowledge-poor approach' (Nielsen 2005: 29). At first, Nielsen (2005) only uses a POS-tagger and no parser, as POS-tagging delivers a 'much higher accuracy than parsing' (Nielsen 2005: 70). This first draft of the detection algorithm takes all auxiliaries as possible candidates to VPE and eliminates false positives by searching the syntactic environment three words before the auxiliary and seven words afterward: 'if it encounters any other verbs, adjectives, nouns, prepositions, pronouns or numbers, [it] classifies the auxiliary as not elliptical' (Nielsen 2005: 71). He later experiments with machine learning techniques, different corpora like the Penn Treebank and the BNC, and automatic parsers to enhance this algorithm's performance.

Obviously, Nielsen (2005) and McShane and Babkin (2015) address a different ellipsis phenomenon than sluices – but this thesis' goal is also distinct from the one of the Santa Cruz Ellipsis Project (Anand and McCloskey 2015a, 2015b). While they built a corpus of sluices – detecting them semi-automatically, this thesis will produce a corpus of wh-final sentences and a fully automatic sluice detecting algorithm.

It can therefore be seen as a pre-step to the Santa Cruz Project or any other sluice corpus creation attempt, facilitating the automatic detection of sluices within wh-final sentences and therefore simplifying and accelerating the process of corpus creation. Additionally, it can serve as the first step in Nielsen's (2005) ellipsis resolution pipeline, specifically tailored to sluices.

4. METHODOLOGY: BUILDING A CORPUS

The wh-finals corpus is built of 479 sentences that contain a sentence-final wh-phrase (see pattern in Figure 7). The data was harvested from a larger corpus and subsequently annotated with significant information.

$$xxxxxx\ xxxxxxx\ xxxxxx\ wh-phrase\ .$$

Figure 7: Schematic representation of wh-final sentences

4.1 Choosing an Approach

Previous projects like the Santa Cruz Ellipsis project (Anand and McCloskey 2015b) used a parsed source corpus (the New York Times subset of the English Gigaword Second Edition corpus) and extracted all verb phrases whose final child was a wh-phrase (Anand and McCloskey 2015a: 3) to obtain sluices. This yielded 5100 instances that were manually assorted: 4100 were actually sluices. McShane and Babkin (2016), in their VPE detection algorithm, extracted auxiliary verbs followed by a punctuation mark or conjunction.

Considering the fact that there is no large open source parsed corpus for German and the approach to detecting sluices was intended to be as barrier-free as possible (i.e. not requiring computationally expensive or costly products) and as automatic as possible (i.e. no manual intervention necessary), a heuristic approach, inspired by the one by McShane and Babkin (2016), was taken. The pattern of 36 German wh-phrases followed by a full stop were pulled from a large plain text corpus: the DWDS core corpora.

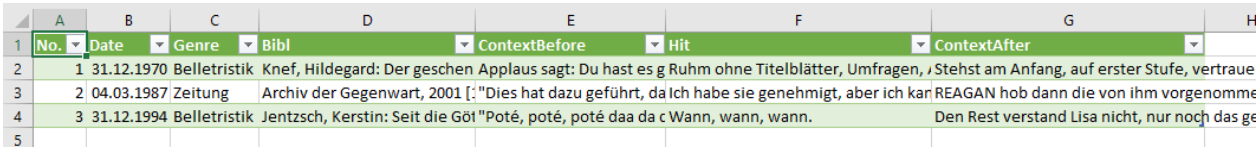
4.2 The Source Corpus

The DWDS (‚Digitales Wörterbuch der deutschen Sprache', *Engl.: Digital Dictionary of the German Language*) (Klein 2018) comprises different dictionaries and nineteen large German text corpora. For the harvesting of data, the two DWDS core corpora 'DWDS Kernkorpus' and 'DWDS Kernkorpus 21' were used: They are balanced (DWDS Kernkorpus) or almost balanced (DWDS Kernkorpus 21) regarding their temporal distribution and distribution over text classes and they are large enough to deliver results for the inquiries. They both contain textual material from newspapers, utility literature, fiction and scientific journals. The DWDS Kernkorpus stretches from 1900-1999, including 120 million tokens of text, the Kernkorpus 21 contains about 15 million tokens from 2000-2010. To limit the size of the composed wh-finals corpus and in order to avoid a diachronic twist to the data, only findings from 1970 or later were allowed (see yellow marking in Figure 8).

The query language for the DWDS is inspired by Regular Expressions: In order to search for 'wann.', the string 'wann' has to be assigned the penultimate position in the sentence ('-2') and combined with ('&&') a final period:

Figure 8: Corpus query: sentence-final 'wann'

The results can be exported as a CSV file. The exported data does not only consist of the sentences including the queried pattern ('Hit'). An index, the publishing date of the text, the genre, bibliographic data (e.g. the author and title of the text or the name of the newspaper), one sentence before and one sentence after the hit are added as additional (meta-)data.



	A	B	C	D	E	F	G
1	No.	Date	Genre	Bibl	ContextBefore	Hit	ContextAfter
2	1	31.12.1970	Belletristik	Knef, Hildegard: Der geschen	Applaus sagt: Du hast es g	Ruhm ohne Titelblätter, Umfragen,	Stehst am Anfang, auf erster Stufe, vertraue
3	2	04.03.1987	Zeitung	Archiv der Gegenwart, 2001 [:	"Dies hat dazu geführt, da	Ich habe sie genehmigt, aber ich kar	REAGAN hob dann die von ihm vorgenomme
4	3	31.12.1994	Belletristik	Jentzsch, Kerstin: Seit die Göt	"Poté, poté, poté daa da c	Wann, wann, wann.	Den Rest verstand Lisa nicht, nur noch das ge
5							

Figure 9: Exemplary output CSV-file

An overview of the data can be found in Table 4:

Corpus	**Kernkorpus**	**Kernkorpus21**	**Sum**
Period	1970-1999	2000-2010	1970-2010
Size of source corpus (sentences)	1.679.394	874.113	2.553.507
Resulting sentences for search patterns	235	254	489
Percentage of wh-final sentences in corpus	0.014%	0.0291%	0.0192%

Table 4: Overview corpus data

The resulting CSV files were merged into one file, the columns 'No.' and 'Date' were deleted.

4.3 The Wh-Finals Corpus

36 search queries containing 36 german wh-phrases were run on the two source corpora. Not all of the wh-phrases led to results, e.g. 'wessen' (*Engl.: whose*) and 'woneben' (*Engl.: next to which)* did not occur with a full stop in the two source corpora.

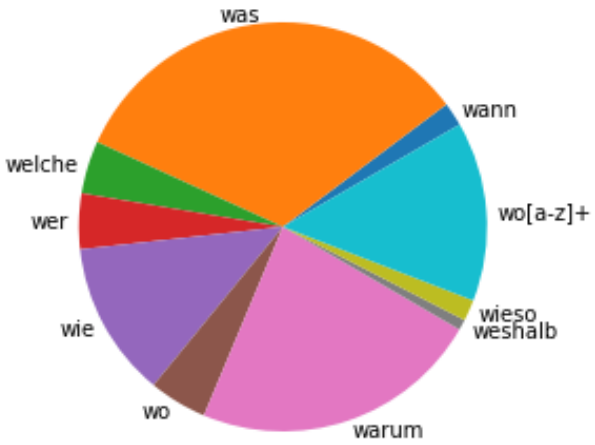


Figure 10: Distribution of wh-phrases in wh-finals corpus

'wo[a-z]' groups together all wh-phrases that begin with 'wo' and incorporate a preposition like 'wofür', 'worüber', 'woran', 'woher' (*Engl.: for what, over what, on what, from what*) etc.

This approach of data harvesting yielded 489 instances of wh-final sentences. The method's downside is that, equivalent to McShane and Babkin (2016), it offers no full recall (McShane and Babkin 2016: 5): not every wh-remnant of a sluice is followed by a full stop (see e.g. sentence-internal sluices, 2.4.2), the method therefore leaves a lot of false negatives behind. Still, a reasonably high number of sluices can be captured this way:

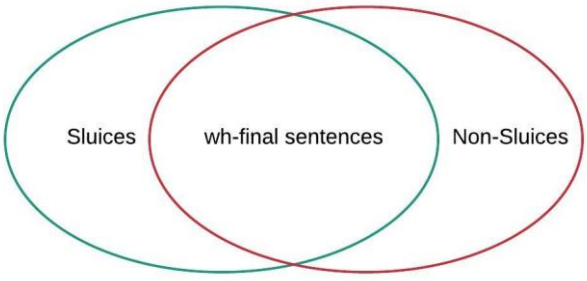


Figure 11: Schematic representation of wh-final sentences in sluices and non-sluices

Out of the 489 results, only 10 had to be culled. They were either repetitions (3 of the results) or parsing errors (7 results) that made the text unsuitable for further processing (see yellow marking in Figure 12).

15: Sloterdijk, Peter: Kritik der zynischen Vernunft Bd. 1, Frankfurt: Suhrkamp 1983, S. 388
Seine Analyse bewahrheitete sich unfreiwillig an ihm selbst.
Allessieht aus **wie.**
Es klingt wie »echt verstanden, ergriffen und gesprochen und ist es im Grunde doch nicht«.

Figure 12: Parsing error

Ungrammatical, but orthographically correct results were left in the corpus and annotated as 'elision' later (see 5.1.1.2).

On the one hand, of course, the corpus was built to enable the development of an algorithm to distinguish between sluices and non-sluices. As a side effect, however, a richly annotated resource for any further corpus-driven research on (German) sluices was created. While, for example, the annotation Merger/Sprouting (see 5.1.3) could not be used in the development of the present algorithm, it showed that the majority of sluices does not have an antecedent – which might have an impact on trying to automatically resolve sluices.

After annotating the corpus, it showed the following distribution of sluice/non-sluice:

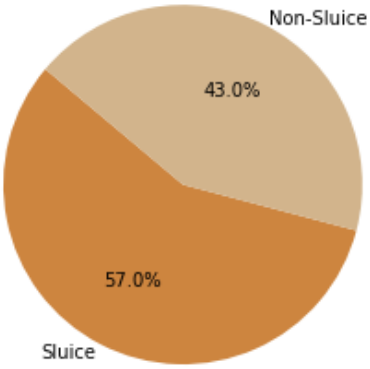


Figure 13: Distribution of sluice/non-sluice in wh-finals corpus

4.4 Additional Corpora

4.4.1 Second Corpus (for Rule-Based Algorithm)

Even though the rules of the algorithm were developed as universally as possible, the result would still be influenced by the data it was developed on (the wh-finals corpus).

For verifying the performance of the rule-based algorithm after its development, a second corpus was built. It consists of 194 wh-final sentences from the 'Berliner Zeitung' (Klein 2018), hand-annotated only with sluice/non-sluice and then processed by the same algorithm as the wh-finals corpus.

This is the distribution between sluice and non-sluice in the second corpus:

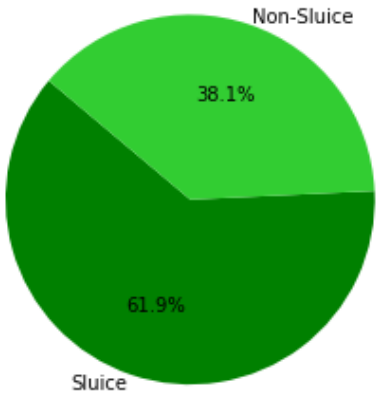


Figure 14: Distribution of sluice/non-sluice in second corpus

4.4.2 Third Corpus (for ML Algorithm)

The performance of the machine learning document classification algorithm was tested on the wh-finals corpus itself, as a function inside the algorithm divides the data into training and test data. However, 'data is the essential resource for any ML project' (Kachkach 2018) and a common conception is: the more data, the better the performance. To investigate if a higher amount of data boosts the ML algorithm's performance, a third corpus, comprised out of the wh-finals corpus and the second corpus, was created, containing 41.6% non-sluices and 58.4% sluices.

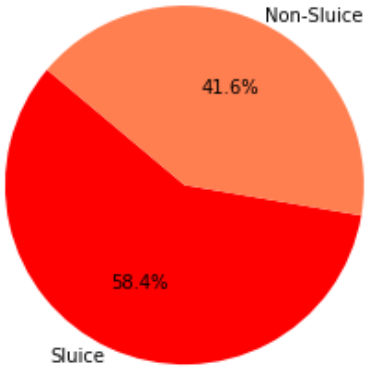


Figure 15: Distribution of sluice/non-sluice third corpus

4.5 Corpora: Overview

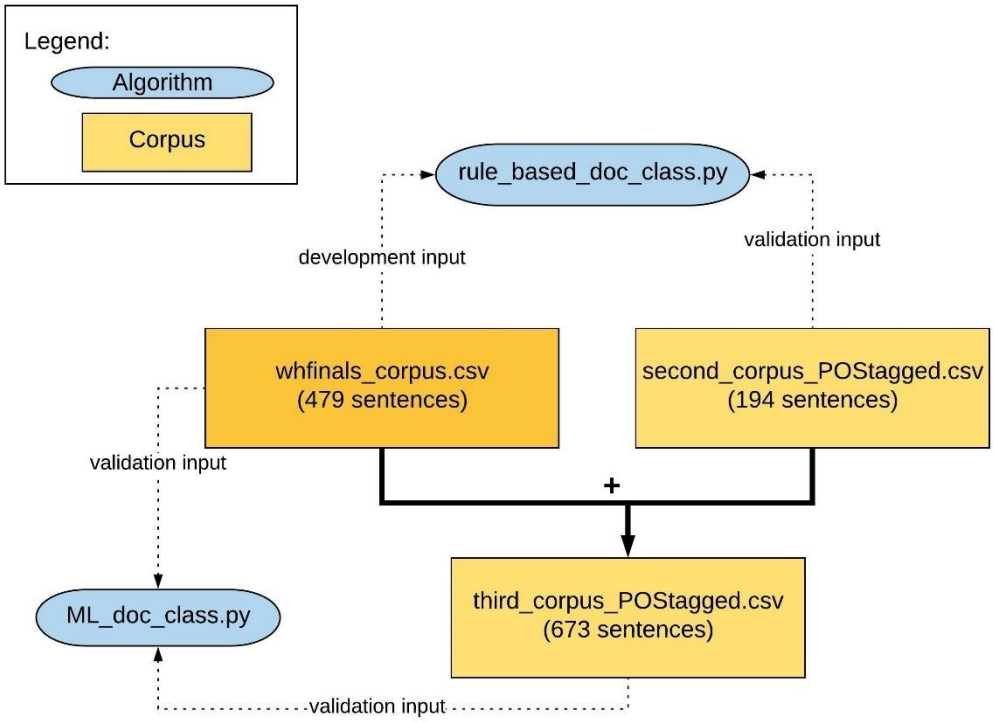


Figure 16: Different corpora and their usage in the algorithms

This flowchart depicts the two approaches to sluice detection and which of the corpora they processed. The rule-based algorithm was developed using the wh-finals corpus and validated by the second corpus. The ML algorithm was run on both the wh-finals corpus and the third corpus.

Note that only the wh-finals corpus has been fully annotated. All findings in chapter 5 rely on this corpus that is most balanced in regards of data tags (sluice/non-sluice), the different wh-words, and thematic resorts.

5. INVESTIGATING THE DATA: LEARNING ABOUT SLUICES

To be able to automatically distinguish sluices from non-sluices, it was crucial to learn about the characteristics of those sentences from real data. Some features could be automatically generated, some had to be hand-annotated.

In addition to the most basic decision (Is the sentence a sluice or not?), facts about the embedding verb, merger and sprouting and the sentence's negation were annotated by hand. The POS-tagged sentences, the wh-phrase at the end of each sentence, its POS and the last verbal element of the sentence were added by a Python-script (`computational_annotation.py`).

5.1 Manual Data Annotation

5.1.1 Sluice or Non-Sluice? (columns G and H)

The sentences in the corpus were annotated as sluice ('y') or non-sluice ('n') which led to the following distribution, as seen in 4.3:

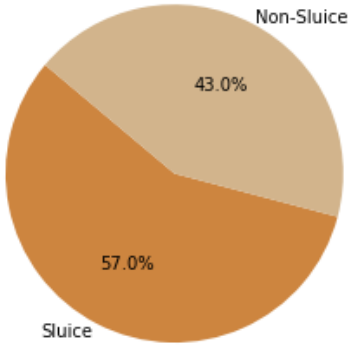


Figure 17: Distribution of sluice vs. non-sluice in wh-finals corpus

Taking a closer look at those 43% of the wh-final sentences that are no sluices (non-sluices), it became clear that they can be divided into four groups:

5.1.1.1 Indefinite Pronouns

A lot of the sentences end in a wh-phrase that does not function as an interrogative phrase but as an indefinite pronoun: the wh-phrase is ambiguous in its POS.

(22)	Von	deinen	Keksen	hab'	ich	auch	noch	welche.
	Of	*your*	*cookies*	*have*	*I*	*also*	*still*	*some.*

5.1.1.2 Elisions

Ungrammatical constructions mostly occur in fictional texts and can serve as stylistic devices. Those sentences were annotated as elisions:

(23)	Ruhm *Fame*	ohne *without*	Titelblätter, *title pages,*	Umfragen, *polls,*	Anfragen, *inquiries,*	Nachfragen, *requests,*
	Geburtsjahr, *year of birth,*	Ort, *location,*	Platz, *place,*	Lieblingsfarbe, *favorite color,*	Lieblingsblume, *favorite flower,*	Lieblingsessen, *favorite food,*
	Sport, *sports,*	Hobby, *hobby,*	Ferien *holidays*	wo *where*	und *and*	wann. *when.*

Some of the sentences in the corpus are incomprehensible, even if their context is considered. It cannot be decided what underlying syntactical structure they have, so they were annotated as elisions, too:

	PREVIOUS CONTEXT				HIT	
(24)	Ich	liebe	zu	wenig.	Aber	wie.
	I	*love*	*too*	*little.*	*But*	*how.*

Missing quotation marks also lead to elisions:

(25)	Ich	meine	ja	dieses	eine	einzige	Wort:	wie.
	I	*mean*		*this*	*one*	*single*	*word:*	*how.*

5.1.1.3 Interjections

Interjections are phrases or words that express emotions like surprise, pain or disbelief. They are not part of a CP, there is no material missing. Therefore, they are no sluices:

	PREVIOUS CONTEXT				HIT	
(26)	Sie	fühlen	sich	ertappt?	Und	wie.
	You	*feel*		*caught?*	*Oh*	*indeed.*

The following interjections were found in the corpus and annotated as such:

- ach was — *oh, come on (annoyed) / oh, really? (ironic)*
- …, was; …, wie — *huh? (expecting agreement)*
- und wie — *oh yes, indeed (strong approval)*
- oder wie — *or what? (disbelieving)*

5.1.1.4 Frozen Expressions

The term ‘frozen expression’ is understood as a generic term for idioms and figures of speech that do not change their shape. Some of them could be seen as sluices, but in most of those cases, the elided IP is not clearly recoverable from the presluice and were therefore annotated as a non-sluice:

(27)	Durch *Due to*	den *the*	Tod *death*	von *of*	Dr. Görne *Dr. Görne*	war *was*	eine *a*	Stelle *position*	frei *free*	geworden,
	und *and*	ich *I*	wollte *wanted*	etwas *something*	festes, *permanent,*	egal, *no matter*	wo. *where.*			

For the sake of consistency, all constructions like these were annotated as Frozen Expressions, even if an elided IP could potentially be found:

	PREVIOUS CONTEXT									HIT	
(28)	Und	wenn	ich	angetrunken	bin,	dann	küsse	ich	gerne.	Egal	wen.
	And	*when*	*I*	*tipsy*	*am,*	*then*	*kiss*	*I*	*like.*	*No matter*	*who*$_{ACC.}$

If the IP in (28) was reconstructed, the sentence would still not be perfectly grammatical (?'…dann küsse ich gerne. Egal wen ich küsse.', *Engl.: ?...then I like to kiss. No matter who I kiss.'*), so the best option seemed a consistent annotation of these expressions as non-sluices. Nevertheless, this classification is arguable.

The following Frozen Expressions were found in the corpus:

- dann und wann	*every now and then*
- oder was	*or what*
- was weiß ich WH-PHRASE	*what do I know WH-PHRASE*
- egal WH-PHRASE, ganz gleich WH-PHRASE, gleichgültig WH-PHRASE, gleich WH-PHRASE	*no matter WH-PHRASE*
- so gut wie	*almost (informal)*
- weiß Gott/der Teufel/ der Himmel WH-PHRASE	*god/devil/heaven knows WH-PHRASE*
- Gott/Wunder weiß WH-PHRASE	*God/wonder knows WH-PHRASE*

Out of the non-sluices, the group of the indefinite pronouns (see 5.1.1.1) clearly is the largest one:

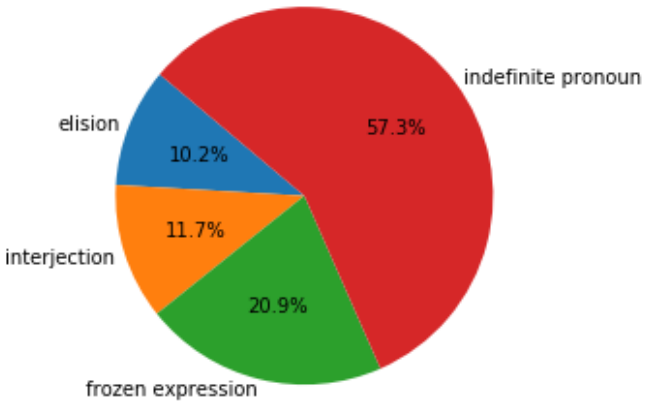


Figure 18: Distribution of non-sluices

5.1.2 Embedding Verb and its Position (columns I and J)

If a sentence turned out to be a sluice, the embedding verb was annotated into column I.

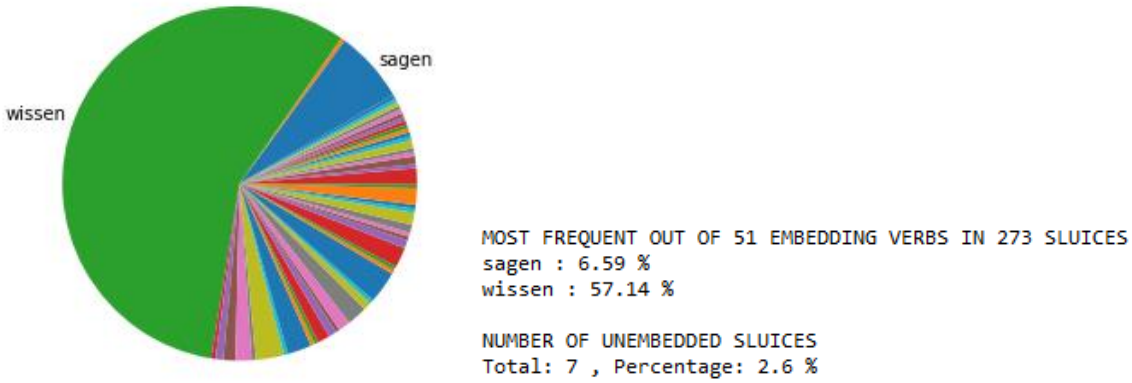


Figure 19: Embedding verbs

It turned out that the 273 sluices in the corpus were embedded by 51 different verbs – or by none: While only 2.6% of the sluices were unembedded (compare to 2.4.6), though, the vast majority was embedded by 'wissen' (*Engl.: to know*), followed by 'sagen' (*Engl. to say*). All other verbs were significantly scarcer.

In the same step, the position of the embedding verb was annotated in column J: if the embedding verb was the last verbal element before the final wh-phrase, column J contains '-1', if it was the penultimate verb, it contains '-2' etc. The greatest distance between embedding verb and wh-remnant was -6 in (29): the embedding functional verb is 'keine Ahnung haben', which is, in this case, followed by five more verbal complexes.

(29)	Ich	**habe**	keine	Ahnung,	wovon	sie	**lebte**,	ob	sie	Geld	**verdiente**
	I	*have*	*no*	*idea*	*of what*	*she*	*lived,*	*if*	*she*	*money*	*earned*
	oder	ob	sie	jemand	**aushielt,**	ob	sie	berufliche	Wünsche	**hatte**,	wohin
	or	*if*	*her*	*someone*	*expensed,*	*if*	*she*	*professional*	*wishes*	*has,*	*where to*
	sie	**wollte**,	und	was.							
	she	*wanted*	*and*	*what.*							

Some instances are clearly sluices but their embedding verb can neither be found in the 'Hit' itself, nor in the 'Context before' (annotated as '<not in context>', see (28)).

	PREVIOUS CONTEXT							HIT	
(30)	Und	wie	Mensch	und	Pflanzen	Wurzeln	schlagen.	Und	warum.
	And	*how*	*humans*	*and*	*Plants*	*roots*	*grow.*	*And*	*why.*

Finally, in some cases, the verb gets elided which leads to a grammatical, albeit colloquial phrase (annotated as '<verb elided>'):

(31)	Für	mich	unerfindlich,	warum.
	For	*me*	*inexplicable*	*why.*

In more than 83% percent of the sluices, the embedding verb is the last verb of the sentence:

```
THESE ARE THE POSITIONS OF THE EMBEDDING VERBS
Position -1: 83.52000000000001%
Position -2: 8.42%
```

Figure 20: Positions of embedding verbs

5.1.3 Merger or Sprouting (column K)

Based on the findings in 2.4.3, the sluices in the corpus were annotated with an 's' in column K if they were sprouting sluices and an 'm' for 'merger' if they had an overt antecedent. 'bef' was added if the antecedent was found in the context before the hit.

»Einem nab ich es schon erzählt.«	Sie brauchte nicht zu sagen, wem.	»Ralph«, stöhnte Davi…	wer	y	nan	sag…	-1	m, bef

Figure 21: Annotation of Merger sluice with antecedent in context before

Similar to Anand and McCloskey (2015a), the wh-finals corpus, too, contains a lot more Sprouting than Merger cases, i.e. a lot more sluices without any antecedent. Anand and McCloskey (2015a) found that only 22% of the sluices in their English corpus have antecedents[17] (Anand and Hardt 2016: 1235). The predominance of Sprouting seems to be a crosslinguistic phenomenon.

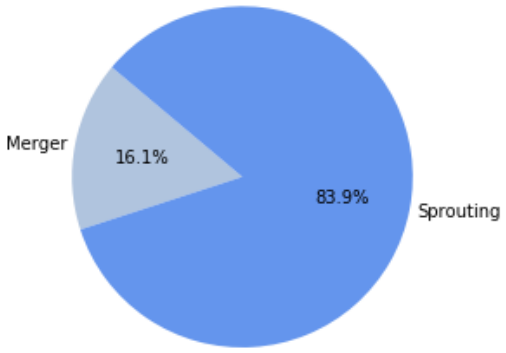


Figure 22: Distribution of Sprouting and Merger in wh-finals corpus

5.1.4 Negation (column L)

Both, sluices and non-sluices were annotated with the information if the clause the wh-word is related to, is negated ('y') or not ('n'). If it is negated, the negating phrase was annotated in the same column (e.g. 'y, indefpron' or 'y, ohne' (*Engl.: without*)). In the corpus, 64% of the sluices were embedded in a negated phrase while only 40% of all sentences were negated:

[17] Note that in their terminology, the antecedent is referred to as 'correlate'.

```
174 of the 273 sluices are negated. (64.0%)

193 of all 479 wh-final sentences are negated. (40.0%)
```

Figure 23: Negated sentences in wh-finals corpus

Sluices are therefore more likely to be negated than the average sentence.

5.2 Computational Data Annotation

The harvested and hand-annotated data was imported into the program (`Step1_computational_annotation.py`) and in one for-loop, iterating over the 479 lines of the corpus, six more columns were added to the CSV-file of the wh-finals corpus.

5.2.1 POS-tagged `Hit` and `Context_before` (column N)

Every `Hit` (sentence containing the final wh-phrase, column D) in the corpus was tagged by the TreeTagger (Schmid 2018). The TreeTagger is a fast and accurate Markov Model tagger that makes use of binary decision trees to estimate transition probabilities (Schmid 1994: 1–2, 1995: 2). It applies the most universally used German tagset: the STTS (Stuttgart-Tübingen-tagset) (Schiller, Teufel and Stöckert 1999).

To be able to execute the TreeTagger onto the corpus in the Python script, a TreeTagger-Python interface was used (Otto 2018). Executed on an Ubuntu 17.10 machine, it facilitates the interaction between Python 3.6, the Natural Language Toolkit 3.3.0 and the TreeTagger 3.2.

The output of the tagger is a list of lists: The lists on the inside consist of the token (Schmid 1995: 8) found in the text, its part of speech, and its lemma: [[token, tag, lemma], [t, t, l], …].

Sie wollte etwas sagen, wusste aber nicht, was. *She wanted to say something, but she didn't know what.*	[['Sie', 'PPER', 'Sie'], ['wollte', 'VMFIN', 'wollen'], ['etwas', 'PIS', 'etwas'], ['sagen', 'VVINF', 'sagen'], [',', '$,', ','], ['wusste', 'VVFIN', 'wissen'], ['aber', 'ADV', 'aber'], ['nicht', 'PTKNEG', 'nicht'], [',', '$,', ','], ['was', 'PRELS', 'was'], ['.', '$.', '.']]

Figure 24: Exemplary tagged hit

5.2.2 wh-Phrase and POS of wh-Phrase (column F and P)

The tagged material was subsequently used to fill column F with the wh-phrase used in the sentence: The sub-list containing the wh-phrase, its POS and its lemma (see yellow marking in Figure 24) was accessed, and the lemma (index 2) extracted. The wh-phrases 'welche' and 'wer' (*Engl.: which, who*) are, as opposed to any other wh-phrase, inflectional (see 2.3), which is the only reason it makes a difference if the lemma or the actual token gets extracted:

```
['welches', 'PWAT', 'welche']
```

Figure 25: POS-tagged token 'welches'

The POS of the wh-phrase (index 1) was put into column O.

It turned out that the sentence-final wh-words were tagged with seven different tags: PWAV (adverbial interrogative or relative pronoun), PIS (substituting indefinite pronoun), PWS (substituting interrogative pronoun), PRELS (substituting relative pronoun), PWAT (attributing interrogative pronoun), KOUS (subordinate conjunction with clause) and KOKOM (comparing conjunction) (Schiller, Teufel and Stöckert 1999).

```
THIS IS HOW OFTEN THE POS-TAGS OCCUR
PWAV occurs 221 times. (31.57%)
PIS occurs 136 times. (19.43%)
PWS occurs 31 times. (4.43%)
PRELS occurs 31 times. (4.43%)
PWAT occurs 1 times. (0.14%)
KOUS occurs 45 times. (6.43%)
KOKOM occurs 14 times. (2.0%)
```

Figure 26: POS-tags of wh-words in wh-finals corpus

5.2.3 Sentence Length (column M)

The length of the sentence was measured and assigned to column M, ensuring punctuation marks are not counted in.

5.2.4 Last Verbal Element and Comparison to Embedding Verb (column Q and R)

Reverse-looping through the text material, the last verbal element before the wh-remnant was extracted and assigned to column P. If no verb could be found in the hit itself, the loop went through the tagged sentence before the hit ('ContextBefore', column C). If it still could not find a verb, one exception was hard coded: if the sequence 'keine Ahnung' (*Engl.: no idea*) was in the hit, the program ascribed 'keine Ahnung haben' to the last verbal element. 'Keine Ahnung', in colloquial contexts, often occurs without the verb 'haben' (*Engl.: to have*). If that was not the case either, '<none>' was put into the column

Subsequently, the last verbal element was compared to the manually annotated embedding verb (column I) to see if they are the same ('y'/'n').

6. DOCUMENT CLASSIFICATION: SLUICE OR NON-SLUICE?

Different NLP projects require different algorithmic approaches. If 'we seek to determine which class(es) or categories a given object belongs to' (Kateb and Kalita 2015: 2), we are facing a document classification task. In this case: it must be decided whether a sentence belongs to the category 'sluice' or 'non-sluice'.

There are a lot of different uses for document classification in the commercial world. Newspaper articles need to be organized by topic or resort, an email provider divides emails into spam or no spam, tweets about a specific topic can be classified into positive, neutral or negative (Li 2018) to enable market analyses. While newspaper documents and most emails are a lot longer than one sentence, Tweets can contain a maximum of 280 characters; the most common length of a Tweet is only 33 characters (Perez 2018). Despite their brevity, Tweets have become one of the most popular data source for all kinds of NLP tasks (document classification, topic modelling, information retrieval etc.) (Kateb and Kalita 2015: 1), which is why the shortness of the wh-final sentences as input for such algorithm was no concern.

6.1 Machine Learning versus Rules

'Machine learning' (ML, henceforth) has developed into a buzzword – while algorithms capable of learning are indeed a major step forward in computational science and information technology (Marr 2016), some say ML is an overrated one-trick-pony, misunderstood as a universal solvent instead of a tool among many (Condliffe 2018; Somers 2017).

Either way, ML is not sorcery: ML algorithms are able to learn the characteristics of preclassified training documents, without being explicitly customized to a specific problem, through the use of statistical methods (Sebastiani 2002: 2). They are, simply put, algorithms that draw conclusions based on large amounts of data (Nicklason 2018; Dorash 2017). Doing so, they do not emulate the decision process of a human expert over a problem, instead, they only take the expert's decisions as a 'ground truth' (Freed 2017) and compare it to the input. The basic proceeding of an ML algorithm is: 'given what we know about these historic outcomes [= training data], what can we say about future outcomes [= test data]' (Parkes 2017)?

Rules, on the other hand, are 'deterministic in nature' (Parkes 2017). Although not as hyped at the moment, rule-based systems do still have their place in data science in general and NLP in particular.

Any rule-based algorithm uses hand-crafted, hard-coded rules in the form of if-then-else statements to make decisions (Parkes 2017; Dorash 2017). A domain expert scans source documents (wh-final sentences, in this case) for rules that help extract key data points (sluice vs.

non-sluice, in this case), always balancing rules that extract too much versus rules that extract too little (Freed 2017).

6.2 Machine Learning Model

In the field of ML, there are two types of approaches: supervised and unsupervised learning (Nielsen 2005: 44). The main difference is that supervised methods require a prior knowledge of what the output will look like: predefined labels. In unsupervised approaches, like clustering, the inherent structure of the data will be depicted without previously determined labels (Soni 2018; Castle 2017). Unsupervised learning algorithms can be useful in some cases as they return categories that a human may not consider (Castle 2017). In the present task though, the categories 'sluice' and 'non-sluice' are clear.

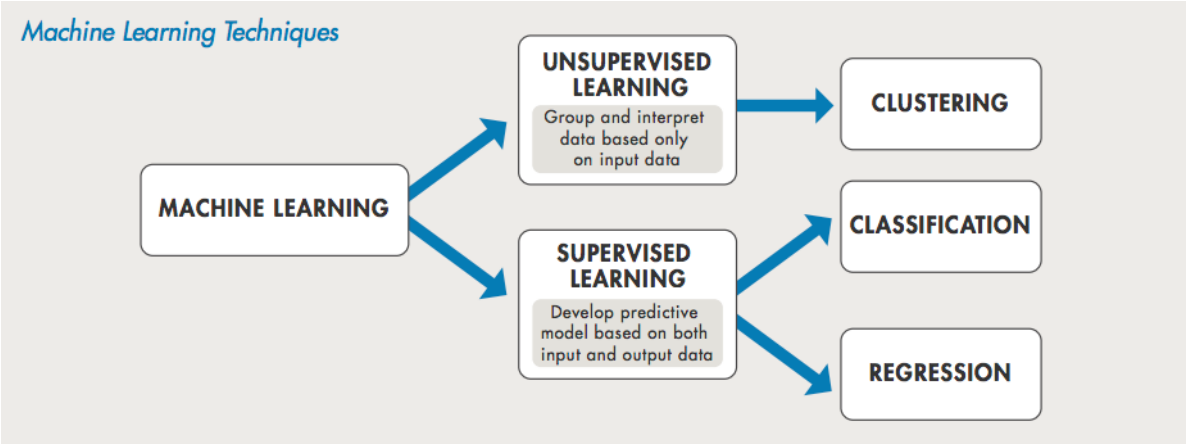


Figure 27: Unsupervised vs. supervised machine learning methods (Kumar 2018)

Typical document classification, in any case, is a supervised method (see Figure 27): an **annotated** dataset containing documents (wh-final sentences) and their labels (sluice/non-sluice) is used to train a 'classifier'. This trained classifier can then divide unlabelled examples (= test data) into the predefined categories (Bansal 2018) 'sluice' and 'non-sluice': it measures the similarity between the given document and the profiles of classes on which it has been trained and then chooses the class(es) to which the object belongs (Kateb and Kalita 2015: 2).

'Supervised learning is so named because the data scientist acts as a guide to teach the algorithm what conclusions it should come up with' (Castle 2017) – the machine can only learn the facts it was taught before. Therefore, falsely labelled input data can only lead to wrong output. In other words, the input the machine is given is the machine's truth, if it's correct or not: computers do not truly understand natural language**.** With the help of machine learning, and the correct input, however, they can get very good at recognizing patterns (Nicklason 2018).

To be able to try out an ML document classification pipeline on the wh-final data, an algorithm developed by Li (2018) was remodelled and adapted to the wh-finals data.

The specific task Li (2018) concentrated on was the following: An incoming consumer finance complaint needs to be classified into one (and only one)[18] of 12 predefined product categories (e.g. consumer loan, mortgage, etc.). The basic structure of Li's (2018) classification algorithm was applied onto the wh-finals corpus as can be seen in the following.

6.2.1 Data Preprocessing

```
1  # -*- coding: utf-8 -*-
2
3  #import all the libraries needed
4  import matplotlib.pyplot as plt
5  import pandas as pd
6  from sklearn.feature_extraction.text import TfidfVectorizer
7  from sklearn.feature_selection import chi2
8  import numpy as np
9
10 from sklearn.model_selection import train_test_split
11 from sklearn.feature_extraction.text import CountVectorizer
12 from sklearn.feature_extraction.text import TfidfTransformer
13 from sklearn.naive_bayes import MultinomialNB
14 from process_stopwords import german_stopwords
15
16 #import csv as dataframe
17 df = pd.read_csv('first_corpus_POStagged.csv',sep=";")
```

Figure 28: Importing libraries (ML approach)

After importing all necessary libraries in the lines 4-14, pandas (`pd`) is used to create a data-frame `df` of the wh-finals corpus (l. 17).

```
19 #change dataframe to only two columns:
20 #independent var Hit and dependent var Sluice
21 col = ['Hit', 'Sluice?']
22 df = df[col]
23
24 #only take those values that are not missing
25 df = df[pd.notnull(df['Hit'])]
26
27 #assign columns to df
28 df.columns = ['Hit', 'Sluice?']
29
30 #adds column category_id, turns y/n into 1/0
31 df['category_id'] = df['Sluice?'].factorize()[0]
32
33 #set up a df that shows all categories
34 category_id_df = df[['Sluice?', 'category_id']].drop_duplicates().sort_values('category_id')
```

Figure 29: Changing dataframe (ML approach)

For a supervised ML document classification, no resources other than the actual text data ('Hit') and the label ('Sluice?') are needed, which is why the dataframe `df` gets reduced to those two

[18] If a document should be allowed in more than one category, a multi-label classification needs to be performed instead. See Nooney (2018).

columns (l. 21-22). Incomplete datapoints are deleted[19] (l. 25) and the column names assigned (l. 28). In line 31, a new column `category_id` is added to `df` which turns the strings 'y' and 'n' into numerical values by applying the pandas method `factorize()`. `factorize()` is 'useful for obtaining a numeric representation of an array when all that matters is identifying distinct values' (pandas 0.23.4 documentation 2018). Line 34 creates a dataframe `category_id_df` that shows all categories in the dataset and their `category_id`. As the wh-finals will only be sorted into two different categories, this dataframe is clear:

Sluice?	category_id
n	0
y	1

Figure 30: `category_id_df` (Output of l. 34)

After these preprocessing and preparation steps, this is what the wh-finals corpus looks like:

Hit	Sluice?	category_id
Ruhm ohne Titelblätter, Umfragen, Anfragen, Nachfragen, Geburtsjah…	n	0
Ich habe sie genehmigt, aber ich kann nicht mehr genau sagen, wan…	y	1
Wann, wann, wann.	n	0
Nicht nur darüber, wer wie viel gespendet hat, herrscht inzwisch…	y	1
Auch in der Politik funktioniert das Heringsprinzip dann und wann.	n	0
Dann und wann.	n	0

Figure 31: Part of the wh-finals corpus after preprocessing

It only contains the wh-final sentence, its annotation saying if it's a sluice or not and the `category_id` based on the annotation ('n' = 0, 'y' = 1).

As is common practice in data science tasks, the initial data is visualized to see if it's reasonably well balanced:

[19] While deleting missing data points is possible and relatively easy, it is by far not the only way of dealing with missing data. There is an ongoing discussion about which way of handling missing or incomplete data points has the least impact on the data. See Swalin (2018).

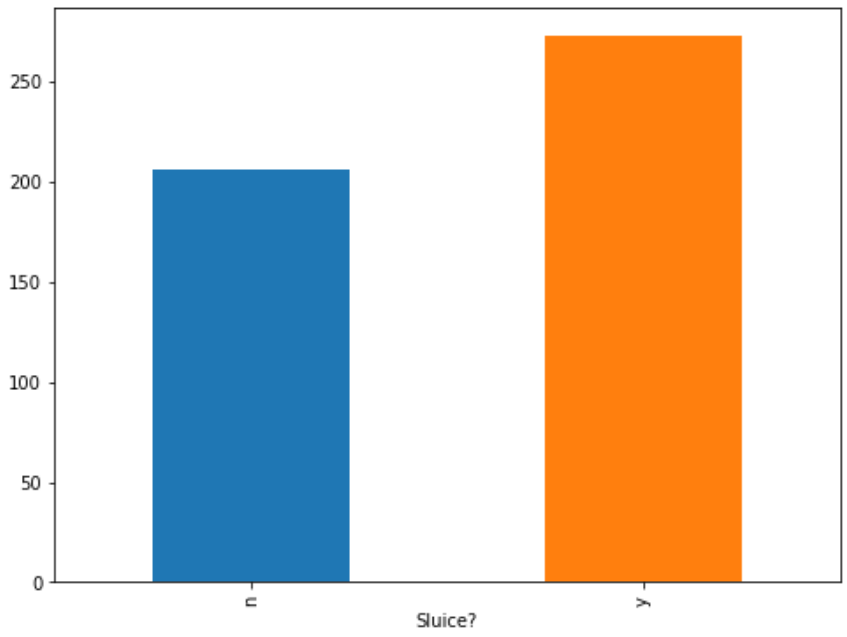


Figure 32: Distribution of non-sluice and sluice in the dataframe

Just like Figure 13, this graph shows that the data is not equally distributed but well enough balanced to perform an ML document classification without the need for rebalancing[20].

6.2.2 Feature Engineering: TF-IDF

6.2.2.1 The Importance of Feature Engineering

'Most of the gains in ML come from great features, not great ML algorithms' (Zinkevich 2016): Sensible feature engineering is the key to success in machine learning applications (Brownlee 2014). For textual data to be useful for the ML algorithm, it needs to be converted into numeric representations – its features need to be transformed.

The data in statistical predictive models consists of two different types of variables: the outcome variable containing data that needs to be predicted (in this case, the tag 'sluice' or 'non-sluice') and **predictor variables** that 'contain data believed to be predictive of the outcome variable' (Bock 2018). A 'feature' in this context is another name for a predictor variable, therefore 'feature engineering' describes the process of creating predictors that can help build a performant predictive model (Bock 2018).

[20] Imbalanced data can be a problem in classification tasks like fraud detection or cancer prediction as the small classes tend to be treated as outliers and ignored. In these cases, the data would need to be artificially rebalanced (Li 2018). There is no definition as of when a dataset is unbalanced, but the general opinion seems to be that a dataset with a 60%/40% or more equal distribution can be considered balanced. As seen in Figure 7, the wh-finals corpus is therefore balanced enough (57%/43%) and does not require further preprocessing.

> 'Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.' (Brownlee 2014)

Most of the classifiers and, in general, learning algorithms in NLP tasks ask for numerical feature vectors with a fixed size instead of a 'free flowing' text (or other) input with variable length (Li 2018; Sarkar 2018).

6.2.2.2 Term Frequency-Inverse Document Frequency (TF-IDF)

In NLP feature engineering, TF-IDF is the 'gold standard of text analysis' (Sandhu 2018). TF-IDF, short for 'term frequency - inverse document frequency', is a way of finding the most valuable classification words for each document. Other than a bag-of-words-model that only counts how many times a word appears in a document (merely term frequency), TF-IDF measures the actual relevance of words (Nicholson 2018).

The model counts the term frequency (TF) and the number of documents the word occurs in (DF). The more often a certain word occurs in all given documents, the less valuable it is as a signal for the character of the specific document. If textual data about a specific topic like movie reviews was investigated, the words 'movie' or 'film' would be discounted – they occur often in the specific document, but they do so in all the documents. That is why they are no valuable signal for the character of the document (Sandhu 2018). TF-IDF effectively identifies a subset of terms that are discriminative for documents in the corpus (Xu *et al.* 2013: 3).

In Li's (2018) code the instance of a TF-IDF vectorizer `tfidf` was built up using scikit-learn's built-in method `TfidfVectorizer` that 'converts raw documents into a matrix of TF-IDF features' (scikit-learn 0.20.2 documentation 2018):

```
45 #initialize tfidf vectorizer
46 tfidf = TfidfVectorizer(sublinear_tf=True, min_df=3, norm='l2', encoding='utf-8', ngram_range=(1, 2))
```

Figure 33: Setup of the TF-IDF Vectorizer (ML approach)

`min_df` builds a threshold for the document frequency: if a word occurs in less than three documents, it will be ignored. `ngram_range` determines 'the lower and upper boundary of the range of n-values for different n-grams to be extracted' (scikit-learn 0.20.2 documentation 2018). It turned out that the `ngram_range=(1, 2)` is ideal for the wh-finals data; the initial `min_df=5` was changed to `min_df=3` in order to yield a better performance of the model.

The TF-IDF vectorizer was initialized excluding German stop words. As there is no built-in list of stop words for German (scikit-learn 0.20.2 documentation 2018) in scikit-learn, a list was pulled from the web (Diaz 2016) and built into the vectorizer (`german_stopwords.txt`). Doing so turned out to diminish the algorithm's performance by 14 percentage points. This leads to

the conclusion that the stop words play a major role in the distinction between sluice and non-sluice: e.g. a preposition preceding the wh-word has an influence on its classification (see 6.3.1.4) and most of the wh-words themselves are listed in the list of stop words. Stop words might even be the predominant differentiating elements of the wh-final sentences. Henceforth, they were not excluded in the analysis.

Note that the TD-IDF model is mainly used in semantic document classification tasks like topic modelling and sentiment analysis where the content of the sentences makes the difference[21]. This is not necessarily the case in sluicing. There shall be no discussion about whether sluicing is a syntactic or a semantic phenomenon here[22]. However, the frequency of words (and the inverse frequency of documents they appear in) is not a merely semantic measure so the application on a syntactic research issue can be just as reasonable.

6.2.3 Model Training

The foundation of every ML predictive model is the separation of the data into training set and test set.

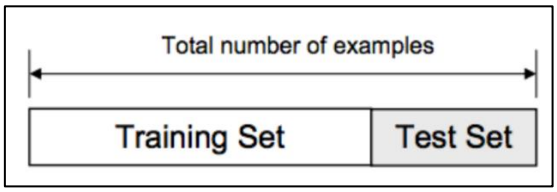


Figure 34: Training and test set (Bronshtein 2017)

In order to deliver statistically correct results, it is crucial that the training set and the test set are kept apart and that the test set does not hold any duplicates of the training set. This can be achieved with scikit-learn's method `train_test_split`. By default, the training set's size is .75 of all examples, the test set is .25 (scikit-learn 0.20.2 documentation 2018).

```
78 X_train, X_test, y_train, y_test = train_test_split(df['Hit'], df['Sluice?'], random_state = 0)
```

Figure 35: Splitting the data into training and test data

[21] Sebastiani (2002) even states that document classification as a whole is an exclusively semantic task: 'Text categorization […], the activity of labelling natural language texts with thematic categories from a predefined set, is [a content-based document management task].' (Sebastiani (2002: 1))
[22] This is a sprawling debate: 'A major reason ellipsis continues to garner such sustained interest is its location on the frontlines of any debate about the division of labor between syntax and semantics.' (van Craenenbroeck and Merchant (2013: 739)). A discussion about the semantic and syntactic identity of sluices can be found in e.g. Chung (2013) and van Craenenbroeck and Merchant (2013).

After the separation of the data, Li's (2018) code explores the performance of four of the most common algorithms in supervised learning (Soni 2018): Logistic Regression, (Multinomial) Naive Bayes, Linear Support Vector Machine, and Random Forest. Applied on the wh-finals data, the following result was obtained:

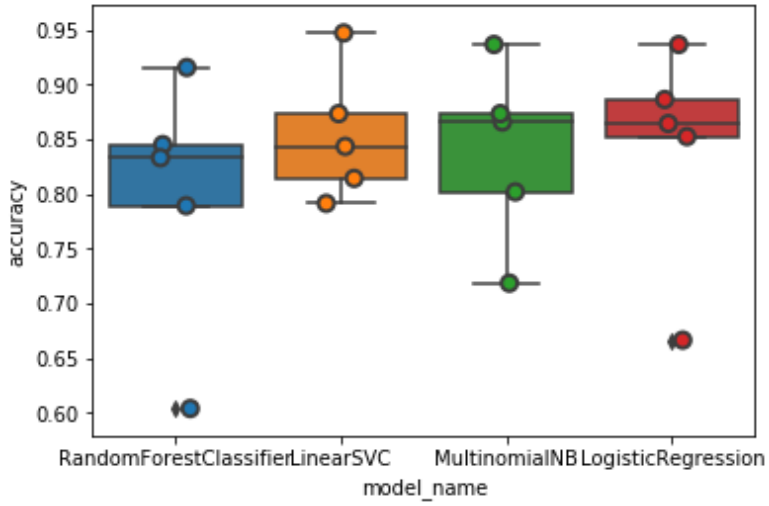


Figure 36: Performance of different statistical models on wh-finals data

While none of the classifiers performs badly, the Random Forest model obviously achieves the lowest result. The following command outputs a readable table of the models' accuracy means for easier comparison:

```
In [26]: cv_df.groupby('model_name').accuracy.mean()
Out[26]:
model_name
LinearSVC                 0.854180
LogisticRegression        0.841464
MultinomialNB             0.839468
RandomForestClassifier    0.797625
Name: accuracy, dtype: float64
```

Figure 37: Comparing statistical models (ML approach)

Just like in Li's (2018) original example, the Linear Support Vector Machine Model (`LinearSVC`) performs best on the wh-finals data and is therefore chosen to be fitted to the test and training data (l. 135-137).

```
134 #decided for LinearSVC (best result)
135 model = LinearSVC()
136 X_train, X_test, y_train, y_test, indices_train, indices_test = \
137 train_test_split(features, labels, df.index, test_size=0.33, random_state=0)
138
139 model.fit(X_train, y_train)
140 y_pred = model.predict(X_test)
```

Figure 38: Training and using the model (ML approach)

The method `fit()` (l. 139) performs the actual training of the classifier: the model is fitted to the training data to enable predictions in the next step (Bronshtein 2017). The core of the algorithm is the application of the `predict()` method in line 140. Corresponding to each wh-final sentence (=`X_test`) in the test set, the model predicts a tag: sluice or non-sluice (=`y_pred`).

6.2.4 Model Evaluation – Wh-Finals Corpus

Li (2018) uses a heatmap to evaluate the performance of her model. To obtain comparable numbers, the calculations of precision, recall and F1-Score were added to the code later.

6.2.4.1 Heatmap and False Classifications

`y_test` and `y_pred` are used to build up a heatmap that depicts the model's success:

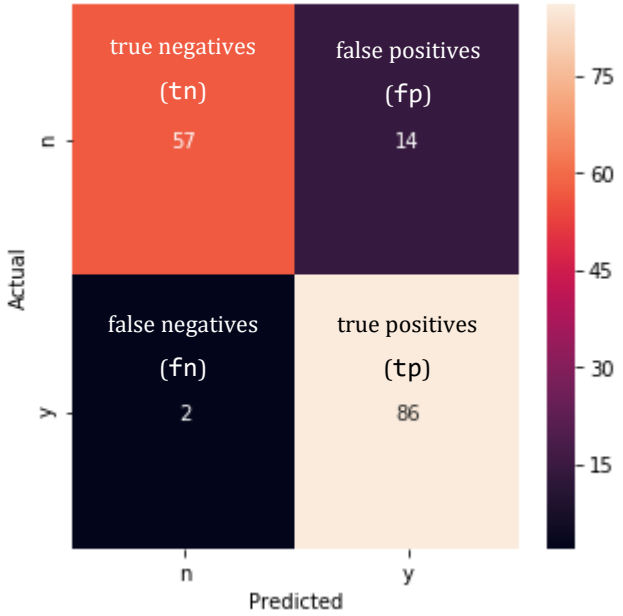


Figure 39: Heatmap on predictions (ML approach, wh-finals corpus)

Eyeballing the graph, one can already see that the absolute values of correct predictions (x='y' and y='y', bottom right; x='n' and y='n', top left) are a lot higher than the false predictions which suggests that this is a high-performance model, albeit no relative metric is given. Besides the heatmap, Li (2018) implemented a display of the falsely predicted documents (in this case, 2+14 sentences). It shows which sentences where predicted to be a non-sluice, even though they are a sluice and vice versa:

```
'y' predicted as 'n' : 2 examples.
   Sluice?                                                  Hit
56       y  Muss wieder daran denken, dass ich vielleicht ...
8        y  Keiner von uns kann sich erklären, warum sie s...

'n' predicted as 'y' : 14 examples.
    Sluice?                                                 Hit
118      n      Doch davor fürchtet sich heute kaum noch wer.
102      n  Als ich am nächsten Tag auf den steinernen Tep...
310      n                                    Warum, warum.
475      n  Niemand kann alles wissen, jeder muss Wissen i...
367      n  Manchmal fragte ich mich, was sie an mir fand,...
351      n         Bei ihm angekommen, essen wir erstmal was.
352      n  Auf Veranstaltungen ans Mikrofon geschubst wer...
90       n  Geradezu eine Schlinge, mit der man dieser "Ro...
461      n  Ich will anerkannt werden als einer, der ihnen...
59       n  »Auf jeden Fall haben sich alle ganz unglaubli...
153      n  Deswegen hatte er dem Taxifahrer auch gesagt, ...
407      n  Wenn etwa Hans Kresnik in seinem Wiener Blut n...
14       n  Die Gründung des Berliner Wissenschaftskollegs...
64       n  »Das hat doch mit Größe nichts zu tun, Mensch,...
```

Figure 40: Falsely predicted documents (ML approach, wh-finals corpus)

6.2.4.2 Precision, Recall, and F1-Score

In order to enable a sensible, number-based evaluation of the document classification code, the performance measures precision, recall and F1-Score of the model were calculated.

$$TP = \text{True positive}$$
$$TN = \text{True negative}$$
$$FP = \text{False positive}$$
$$FN = \text{False negative}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 41: How to calculate precision, recall and F1-Score (Hui 2018)

Precision and recall express different perspectives on the performance of a statistical model.

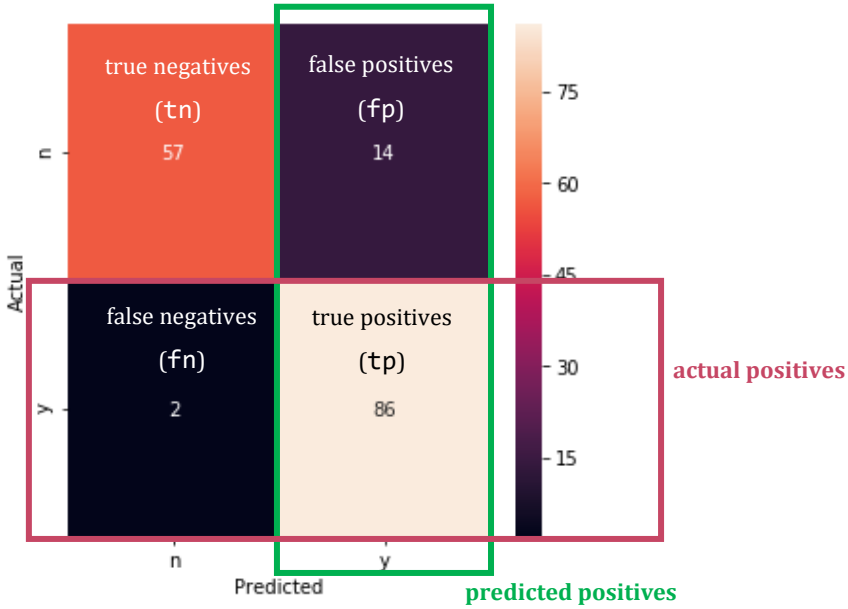


Figure 42: Four values needed for the calculation of precision and recall

a) How many selected items are relevant?: Precision

The precision value divides the `tp` through the total number of 'predicted positives' (`fp+tp`). It therefore describes how many of the positive predictions are indeed positive. Precision should be used if `fp` have to be avoided as much as possible. This is the case in spam detection: an email predicted as spam and put into the spam folder even though it is no spam might result in the loss of important information (Shung 2018; Hui 2018).

b) How many relevant items are selected?: Recall

The recall measure asks the question: How many of the 'actual positives' are captured? It calculates how many of the 'actual positives' the model could find. Equivalent to the logic above, the recall should be used if the cost of an `fn` is very high. This is the case in sick patient prediction: If a sick person (actual positive) gets predicted as negative (and therefore ends up being a false negative), the cost could be very high (e.g. if the sickness is life threatening or contagious) (Shung 2018).

If it is not clear whether `fp` or `fn` are worthier being avoided or from which perspective the performance should be measured, the **F1-Score** can be used. It builds the harmonic mean between precision and recall (Cirić 2018).

Precision, recall and F1-Score were computed by comparing the y-values of the test data (i.e. if the sentence is a sluice or not) with the y-values that were predicted by the algorithm and the following results were obtained:

```
In [88]: from sklearn.metrics import f1_score, precision_score, recall_score
    ...:
    ...: print("Precision:",round(precision_score(y_test, y_pred),4))
    ...: print("Recall:",round(recall_score(y_test, y_pred),4))
    ...: print("F1-Score:",round(f1_score(y_test, y_pred),4))
Precision: 0.86
Recall: 0.9773
F1-Score: 0.9149
```

Figure 43: Precision, recall and F1-Score (ML approach, wh-finals corpus)

As there are only 2 `fn` (see Figure 42), the recall of the ML model is extraordinarily high. This means, out of all the sluices in the corpus, 97.73% were captured. The precision value is lower, which means that some of the sentences captured were actually not sluices. All in all the ML document classification algorithm by Li (2018) applied onto the wh-finals data yielded an F1-Score of 91.49%. In other words, the algorithm, after learning about the data, can predict sluice/non-sluice for unseen data with an accuracy of 91.49%.

6.2.5 Model Evaluation – Third Corpus

It is a common assumption that ML algorithms need large amounts of data. While (for different reasons) there is no resource to be found that states an exact number of examples a dataset should have to be a good input for an ML algorithm, 479 data points in the present corpus seem to be a very small dataset. Popular datasets, for e.g. sentiment detection, like the IMDB Reviews corpus[23] or the 140Sentiment corpus[24] hold thousands or even millions of examples.

6.2.5.1 Heatmap and False Classifications

After enlarging the wh-finals corpus by the second corpus, resulting in the third corpus, it contains 673 wh-final sentences which were classified by the ML algorithm in the following manner:

[23] See http://ai.stanford.edu/~amaas/data/sentiment/
[24] See http://help.sentiment140.com/

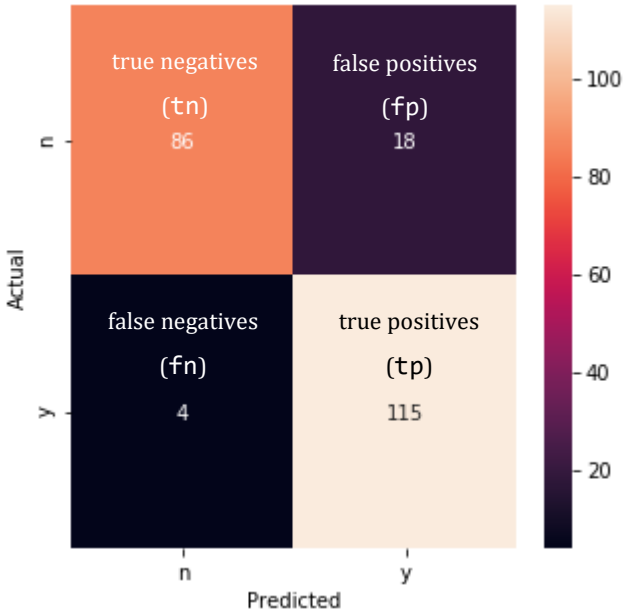


Figure 44: Heatmap on predictions (ML approach, third corpus)

The 22 false classifications were listed, too:

```
'y' predicted as 'n' : 4 examples.
    Sluice?                                                 Hit
85        y                                Oder wer ich bin; was.
245       y  Auch wer abgelehnt wurde, weiß dann wenigstens...
547       y  Klar, beim Zaubern passiert schon mal was, abe...
637       y  ---- 5. Februar Immer wieder begegnen einem di...

'n' predicted as 'y' : 18 examples.
    Sluice?                                                 Hit
14        n  Die Gründung des Berliner Wissenschaftskollegs...
404       n  Jetzt sitzen sie, mal zwölf, mal drei, mal fün...
339       n  Wolfgang, dem nicht so leicht etwas entging, s...
426       n  Know how kommt als Fachausdruck aus dem amerik...
584       n  Ein Skandal sei es beispielsweise, daß das Lan...
585       n  Frau Goschev setzt sich in den Aufenthaltsraum...
424       n  Nächster Vorschlag: (ich hatte ein Exemplar de...
310       n                                   Warum, warum.
336       n  Gewiß, wir können Fehler machen, und wenn Sie ...
534       n                                    Gewußt wie.
618       n                              Sowieso, weshalb, wozu.
453       n  Blechern aber kann ich mir keinen Laut vorstel...
649       n  Nicht alles, was aussieht wie Müll, ist auch w...
443       n  Wenigstens blühen im Gemüsegarten die Bohnen, ...
391       n  Er gab sich auch nicht wie ein Schweizer, irge...
561       n                                           I wo.
511       n  Der Berliner Arbeitsrichter Friedbert Rancke w...
97        n  Eine Geschichte, ein Roman, ein Märchen " dies...
```

Figure 45: Falsely predicted documents (ML approach, third corpus)

Note that these 22 false classifications are very different from the false classifications of the first corpus, even though the whole first corpus is part of the third corpus. Only two of the false classifications in the first corpus can be found in the false classifications of the third corpus.

6.2.5.2 Precision, Recall, and F1-Score

If more data led to a better algorithm performance, this slightly larger dataset should yield better than or at least similar results to the first dataset. This was not the case:

```
In [38]: from sklearn.metrics import f1_score, precision_score, recall_score
    ...:
    ...: print("Precision:",round(precision_score(y_test, y_pred),4))
    ...: print("Recall:",round(recall_score(y_test, y_pred),4))
    ...: print("F1-Score:",round(f1_score(y_test, y_pred),4))
Precision: 0.8647
Recall: 0.9664
F1-Score: 0.9127
```

Figure 46: Precision, recall and F1-score (ML approach, third corpus)

While, compared to the smaller original corpus (see Figure 43), the precision, meaning the number of selected items that were relevant, grew slightly, the recall dropped slightly. Altogether results in an F1-Score of 91.27% which is marginally lower than the F1-Score of the algorithm on the original dataset (91.49%).

This leads to the conclusion that a larger dataset does not automatically enhance an ML document classification algorithm's accuracy.

6.3 Rule-Based Model

While the ML document classification merely required the wh-final sentences and their tags (sluice/non-sluice) as input, the rule-based document classification algorithm uses the information that was annotated before (see 5.2). To keep the algorithm as universal as possible and truly automatic, only the automatically annotated meta-data was used. The handmade annotations led to numerous findings that helped in the development of the pipeline but were not utilized in the classification algorithm itself.

The goal of this algorithm is to separate the non-sluices from the sluices. As already pointed out in 5.1.1, the non-sluices can be distinguished into four groups: indefinite pronouns, elisions, interjections and frozen expressions. The latter two could mostly be identified using Regular Expressions, while elisions and indefinite pronouns were not as easily singled out.

6.3.1 Rules

In the following, every rule of the algorithm is described. Their respective performance was measured while putting them as first rule in the algorithm. The structure of each rule is the

same: if a certain statement is true, the sentence will be predicted as sluice or as non-sluice, depending on the rule. The prediction 'y' (sluice) or 'n' (non-sluice) will be put into column Q of the CSV-file. For measuring the performance of the algorithm later, a script for calculating precision, recall and F1-score was developed (`precrec.py`), comparing column Q (prediction) to column G (sluice or non-sluice).

6.3.1.1 Rule 1: Check for Fixed Phrases (`regexes`)

As the name indicates, frozen expressions are fixed phrases that always occur in the same shape. The same applies to the interjections listed here. This makes them predetermined for the identification through Regular Expressions.

The following tuples, containing Regular Expressions, were set up:

```
18 #frozen expressions:
19 #wer weiß w, Gott weiß w, weiß Gott w, ganz gleich w, egal w, dann und wann, gleichgültig w
20 #weiß der Teufel w, der Teufel weiß w, ganz gleich w, Wunder weiß w, gewusst wie, weiß der Himmel w,
21 #so gut wie, was weiß ich w
22 frozen_expressions = ('wer weiß\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
23                       'gott weiß\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
24                       'weiß gott\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
25                       'egal\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
26                       'dann und wann',\
27                       'gleichgültig\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
28                       'weiß der teufel\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
29                       'der teufel weiß\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
30                       '(ganz)? gleich\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
31                       'wunder weiß\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
32                       'gewu[sß]+t wie\.',\
33                       'weiß der himmel\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.',\
34                       'so gut wie\.',\
35                       'was weiß ich\,?\s?(\w*)? w[a-zäöü]+( viel(e?))?( und\s?(\w*)? w[a-zäöü]+)?\.')
36
37 #interjections:
38 #ach was, na sowas, und wie
39 interjections = ('[^A-Za-z]?ach\,? w[a-z]+\.', 'na so ?was\.', '^und wie\.', '\, und wie\.')
40
41 #hint for 'was' to be a pronoun: so was (short for 'so etwas')
42 hint_for_indefpron = ('so was\.', '\, so was\.')
```

Figure 47: Tuples of Regular Expressions

The Regular Expressions for frozen expressions were designed so that they cover every possible wh-word and multiple wh-words (compare to 2.4.4), e.g. 'wer weiß, wie.' (*Engl.: who knows how*) (l. 22) or 'weiß der Teufel warum und weshalb.' (*Engl. the devil knows why*) (l. 28).

In 5.1.1 it was found that, out of the 206 non-sluices in the corpus, 67 (24 interjections + 43 Frozen Expressions) were fixed expressions:

```
Out of the 206 nonsluices, there are
21 annotated as elision
24 annotated as interjection
43 annotated as frozen expression
118 annotated as indefinite pronoun.
```

Figure 48: Distribution of non-sluices over non-sluice categories

In addition to the frozen expressions and interjections, it turned out in the corpus that in the expression 'so was' (*Engl. something like that*) 'was' is always an indefinite pronoun and never an interrogative pronoun, so it could be hard-coded too (l. 42).

The following function `nonsluice_finder`

```
'''1. Checks if a fixed expression is in the sentence'''
'''If yes, predicted as non-sluice'''
############################################################
#function to determine if non-sluice is in row
def nonsluice_finder(which_non, input_row):
    hit=str(input_row[3])
    for non in which_non:
        #apply RE on hit-column, lower space it
        non_detect = re.search(non, hit.lower())

        #returns True if RE is found in sentence
        if non_detect:
            return True
```

Figure 49: Function to apply Regular Expressions

outputs `True` if it finds one of the Regular Expressions in one of the sentences in the corpus.

Subsequently, the function `regexes` is applied with all three of the Regular Expression tuples (Figure 50 exemplarily shows the part for frozen expressions).

```
102         if regexes.row[18] != 'y' and regexes.row[18] != 'n':
103             if nonsluice_finder(frozen_expressions,regexes.row):
104                 regexes.row[18]='n'
```

Figure 50: Rule 1 - `regexes`

Line 102 makes sure that only cases that have not been tagged yet, will be treated[25]. If the Regular Expression is found in the sentence (l. 103), `row[18]`, the prediction column of the dataframe, will be filled with 'n' – non-sluice (l. 104).

In the same step the algorithm checks the impact of the rule and what the findings were:

```
22 found as interjections,
42 found as frozen expressions,
38 found as indefinite pronoun

2 false prediction(s) by regexes:
- Ob nun Schwarm, Ableger oder Volk: Kaufen Sie niemals Bienen ohne die Seuchenfreiheitsbescheinigung, die
Ihnen gesunde Völker garantiert und die außerdem vorgeschrieben ist, wenn Sie Ihre Bienen beziehen,
gleichviel woher.
- »... und ich sage dir gleich, warum.
100 correct prediction(s).
```

Figure 51: Impact of `regexes`

[25] This line is part of every rule in the algorithm.

102 sentences were classified as 'n' out of which 100 were actual non-sluices (=correct predictions) and only two sentences were falsely tagged. `regexes` therefore is a very successful classification rule.

*6.3.1.2 Rule 2: Check the POS of Wh-Words (*POS*)*

A function was implemented that yielded the following overview of the sentence-final wh-words in the corpus and their POS:

how often	wh-phrase	Non-Sluice	Sluice
9	wann	{'PWAV': 4}	{'PWAV': 5}
158	was	{'PIS': 119, 'PWS': 5, 'PRELS': 5}	{'PRELS': 20, 'PWS': 5, 'PIS': 4}
20	welche	{'PIS': 11}	{'PIS': 2, 'PRELS': 6, 'PWAT': 1}
21	wer	{'PWS': 8}	{'PWS': 13}
59	wie	{'KOUS': 17, 'KOKOM': 6}	{'KOUS': 28, 'KOKOM': 8}
22	wo	{'PWAV': 12}	{'PWAV': 10}
1	wobei	{}	{'PWAV': 1}
5	wofür	{}	{'PWAV': 5}
9	woher	{'PWAV': 2}	{'PWAV': 7}
23	wohin	{'PWAV': 4}	{'PWAV': 19}
9	worüber	{'PWAV': 3}	{'PWAV': 6}
9	wozu	{}	{'PWAV': 9}
110	warum	{'PWAV': 7}	{'PWAV': 103}
4	weshalb	{}	{'PWAV': 4}
8	wieso	{}	{'PWAV': 8}
1	wodurch	{}	{'PWAV': 1}
1	womit	{'PWAV': 1}	{}
1	woran	{}	{'PWAV': 1}
5	worauf	{}	{'PWAV': 5}
4	wovon	{'PWAV': 2}	{'PWAV': 2}

Figure 52: Overview over wh-words and their POS

First, the output of the TreeTagger cannot be considered correct. If it was, all wh-words in sluices (right side of the table) would be tagged as an interrogative pronoun (PWAV, PWS or PWAT[26] (see 5.2.2)) as all sentences listed in the right column are elided questions. Instead, PRELS, PIS, KOUS and KOKOM are found as well. These outcomes can be seen as another proof for the difficulties of machines (in this case, the TreeTagger) dealing with ellipses.

Second, even if the meaning of the tags was ignored, there is no clear difference between the POS of wh-words in sluices and in non-sluices (e.g. 'wer (*Engl.: who*) is tagged as PWS no matter if the sentence is a sluice or not). Only 'was' and 'welche' (*Engl.: what*, *which*) are likely to be non-sluices if they are tagged as PIS (substituting indefinite pronoun). The tag, in this case,

[26] Reminder:
PWAV = adverbial interrogative or relative pronoun,
PWS = substituting interrogative pronoun,
PWAT = attributing interrogative pronoun,
PRELS = substituting relative pronoun,
PIS = substituting indefinite pronoun,
KOUS = subordinating conjunction with clause,
KOKOM = comparing conjunction (Schiller, Teufel and Stöckert (1999))

makes sense: PIS stands for a type of indefinite pronouns, which are the largest group in the non-sluices.

In all other cases, the data is too sparse to draw conclusions from it. For 'was' and 'welche' though, this information was transferred into a function:

```
'''2. POS: Utilize the POS we can'''
#############################################################################################
def POS(which_df):
    instant_vars(POS)
    for POS.index, POS.row in which_df.iloc[1:].iterrows():
        if POS.row[18] != 'y' and POS.row[18] != 'n':

            if POS.row[5]=='was':   #check if 'was' is POS-tagged as 'PIS'
                if POS.row[15]=='PIS':
                    POS.row[18]='n'     #if so, predict as non-sluice
                    check_predictions_nonsluice(POS)
                else:
                    POS.row[18]='y'    #if 'was' is tagged as anything else, predict as sluice
                    check_predictions_sluice(POS)

            if POS.row[5]=='welche':   #check if 'welche' is POS-tagged as 'PIS'
                if POS.row[15]=='PIS':
                    POS.row[18]='n'    #if so, predict as non-sluice
                    check_predictions_nonsluice(POS)
                else:
                    POS.row[18]='y'    #if 'welche' is tagged as anything else, predict as sluice
                    check_predictions_sluice(POS)

    summary_predictions(POS)

#############################################################################################
```

Figure 53: Rule 2 - POS

The code picks out 'was' and 'welche' and assigns 'n' into their prediction column if they carry the POS-tag 'PIS'; in case they don't carry 'PIS' they are tagged as sluices ('y').

In this table, the impact of the rule can be retraced:

```
16 false prediction(s) by POS:
- Ich meine, schlabbern die schon um die Knie herum, oder was.
- Ach, was.
- »Ach was.
- Trinken, dachte David, egal, was.
- »Ach was.
- Mein Onkel sagt zu meinem Vater, zwei zu eins ist nicht schlecht, was.
- »Ja, aber was für welche.
- Erstens: Ich wußte nicht, wo Sofie war, was sie tat, ob sie in Gefahr schwebte oder nicht, und wenn, in
welcher.
[...]
162 correct prediction(s).
```

Figure 54: Impact of POS

*6.3.1.3 Rule 3: Check for wo[a-z]+ and why (*woAZ_why*)*

Unrelated to the POS, the table led to a second realisation: In an interim development step the code was run after the application of the Regular Expression, and showed the following:

how often	wh-phrase	NO sluice	YES sluice
6	wann	{'PWAV': 1}	{'PWAV': 5}
101	was	{'PIS': 68, 'PWS': 2, 'PRELS': 2}	{'PRELS': 20, 'PWS': 5, 'PIS': 4}
20	welche	{'PIS': 11}	{'PIS': 2, 'PRELS': 6, 'PWAT': 1}
20	wer	{'PWS': 7}	{'PWS': 13}
44	wie	{'KOKOM': 1, 'KOUS': 7}	{'KOUS': 28, 'KOKOM': 8}
12	wo	{'PWAV': 2}	{'PWAV': 10}
1	wobei	{}	{'PWAV': 1}
5	wofür	{}	{'PWAV': 5}
7	woher	{}	{'PWAV': 7}
20	wohin	{'PWAV': 1}	{'PWAV': 19}
6	worüber	{}	{'PWAV': 6}
9	wozu	{}	{'PWAV': 9}
106	warum	{'PWAV': 4}	{'PWAV': 102}
4	weshalb	{}	{'PWAV': 4}
8	wieso	{}	{'PWAV': 8}
1	wodurch	{}	{'PWAV': 1}
1	woran	{}	{'PWAV': 1}
5	worauf	{}	{'PWAV': 5}
2	wovon	{}	{'PWAV': 2}

Figure 55: POS on new data

With most of the frozen expressions, interjections and some of the indefinite pronouns taken away from the data (compare to Figure 52), it shows that only one out of of all sentences ending in a wo[a-z]+ is a non-sluice, so sentences with wo[a-z]+ can be generalized as sluices. The different versions of 'why' ('wieso', 'weshalb', 'warum', 'weswegen') were generalized accordingly as they also lean far more in the direction of sluices.

```
161             #check if whphrase is a wo[a-z]+ or a version of 'why'
162             if woAZ_why.row[5] in list_woAZ or woAZ_why.row[5] in ['warum','weshalb','wieso', 'weswegen']:
163                 woAZ_why.row[18]='y'    #if so, predict as sluice
```

Figure 56: Rule 3 – woAZ_why

As Figure 55 suggested, the generalisation of sentences with 'wo[a-z]+' and 'why' is highly successful.

```
19 false prediction(s) by woAZ_why:
- Egal woher.
- Egal woher.
- Ich drehe Cems Ring an meinem Finger und gehe wieder mit ihm, egal wohin.
- Egal wohin.
- Es waren bemitleidenswerte Kreaturen aus der Gattung der Doppelwesen, die aus einem sprechenden Kopf und einem
sprechenden Bauch bestanden, die sich eigentlich immer uneins waren, egal worüber.
[...]
171 correct prediction(s).
```

Figure 57: Impact of woAZ_why

6.3.1.4 Rule 4: Check for prepositions (prep_check)

If the sentence-final wh-word is preceded by a preposition, the sentence is more likely to be a sluice (compare to 2.4.7). Here, the third last element of the sentence was accessed (line 180) (second last element is the wh-word and last is the fullstop) and its POS compared to all possible preposition tags (APPR and APPRART).

```
179             tagged=list(prep_check.row[13])
180             mini_list=tagged[-3] #access third last element of sentence
181             if mini_list[1] in tags_prepositions: #see if the third last element is a preposition
182                 prep_check.row[18]='y'#if it is a preposition, tag sentence as sluice
```

Figure 58: Rule 4 - prep_check

This was an intuition confirmed by the data: There can be no grammatical sentence with a PREP+WH-pattern in the end that is no sluice. The only sentences that were falsely tagged as sluice here were ungrammatical elisions:

```
3 false predictions by preposition_check:
- Befreiung von was.
- - Zum Aufheben für wen.
- Wirr um was.
10 correct predictions.
```

Figure 59: Impact of prep_check

6.3.1.5 Rule 5: Check the last Verbal Element (`embedding`)

In this step of the algorithm, the last verbal element of the sentence was investigated. In 5.1.2 it was found that in 83.5% of the cases, the embedding verb of a sluice is the last verbal element in the sentence.

In `embedding`, `row[16]`, which contained the last verbal element of the sentence, was compared to the list `licensing_verbs` that was created specifically for this task:

```
311 '''VI. verb_list: List of QEV + function to add all the licensing verbs in the corpus'''
312 ##############################################################################################
313 licensing_verbs = ['wissen', 'erinnern', 'vergessen', 'ankündigen', 'bemerken', 'lernen', 'herausfinden',
314                    'entdecken', 'erzählen', 'zeigen', 'aufzeigen', 'anzeigen', 'informieren', 'preisgeben',
315                    'entscheiden', 'festlegen', 'spezifizieren', 'einigen', 'kontrollieren', 'erraten',
316                    'vorhersagen', 'wetten', 'schätzen', 'keine Ahnung haben', 'sicher sein', 'unsicher sein',
317                    'fragen', 'untersuchen', 'interessieren', 'nachfragen', 'herausbekommen', 'angeben',
318                    'begründen', 'erklären', 'schildern', 'stellen', 'stehen', 'bekanntgeben', 'zeigen'
319                    'raten', 'entscheiden', 'herausbekommen', 'merken', 'lauten']
320
321 def verb_list (which_df, which_list):
322     i=0
323     for index, row in which_df.iloc[1:].iterrows():
324         embed_verb=row[8]
325         if embed_verb not in which_list and '<' not in str(embed_verb):
326             which_list.append(embed_verb)       #extend verblist with verbs occuring in the corpus
327             i+=1
328     print(i,'verbs added to the list of QEV (question embedding verbs).\n\n\n')
```

Figure 60: Function to build a list of QEV

The list `licensing_verbs` initially (l. 313-319) holds most of the verbs proposed as QEV by Lahiri (2002) and Karttunen (1977) (see Figure 5 and chapter 2.5) translated to German. In line 323-326, all the embedding verbs occurring in the wh-finals corpus are added to the list. It should still not be considered comprehensive.

If the last verb of the sentence was found in `licensing_verbs`, the sentence was annotated as sluice.

```
198             #check if the last verb is a QEV
199             if embedding.row[16] in licensing_verbs:
200                 embedding.row[18]='y'     #if so, predict as sluice
```

Figure 61: Rule 5 – embedding

`embedding` produces 30 false positives, but also 196 true positives.

```
30 false prediction(s) by embedding:
- Tante Else eine messerscharfe Idee mitteilen, ach was.
- Dabei werden sie hoffärtig, der Statusunterschied wird zum Problem, sie fallen vom Dichter ab und halten sich
selber für Gott weiß was.
- Befreiung von was.
- Es gibt mehr Kneipen als Kirchen oder Galerien oder Konzerthäuser oder Clubs oder Discos oder was weiß ich was.
- »Herr Lehmann, ich sag dir mal was.
[...]
196 correct prediction(s).
```

Figure 62: Impact of embedding

6.3.1.6 *Rule 6: Check if Wh-Word can be Pronoun (not_pronoun)*

As the biggest group of non-sluices are indefinite pronouns (see Figure 18) it seemed reasonable to implement a function that checks if the wh-word can be an indefinite pronoun at all:

```
216             cant_be_pron=['wieso', 'weshalb', 'warum', 'weswegen', 'wann', 'wie', 'wo']
217             if not_pronoun.row[5] in cant_be_pron:  #check if wh-word is a not-pronoun
218                 not_pronoun.row[18] = 'y'   #if so, predict as sluice
```

Figure 63: Rule 6 – not_pronoun

If it cannot be a pronoun, the sentence is annotated as a sluice (l. 218).

Just like `embedding`, `not_pronoun` can fetch a lot of the sentences and therefore generates some false positives, but also a lot of true positives.

```
46 false prediction(s) by not_pronoun:
- Ruhm ohne Titelblätter, Umfragen, Anfragen, Nachfragen, Geburtsjahr, Ort, Platz, Lieblingsfarbe, Lieblingsblume,
Lieblingsessen, Sport, Hobby, Ferien wo und wann.
- Wann, wann, wann.
- Auch in der Politik funktioniert das Heringsprinzip dann und wann.
- Dann und wann.
- Da kommen die Leute aus Alabama und Texas und wollen einfach nur einen Hirsch abknallen, egal, wie.
- Und wie.
[...]
166 correct prediction(s).
```

Figure 64: Impact of not_pronoun

6.3.1.7 *Rule 7: Check if Punctuation is in Last Elements (*`punctuation`*)*

The orthographically correct version of every German sluice holds a comma (or a different punctuation mark in the same function) before the elided CP (see 2.3). It can stand directly before the wh-word (see (32)) or further inside the sentence (see (33)):

(32) Dann hatte das aufgehört, ohne daß er noch wusste, wann.

	Then	*had*	*this*	*stopped*	*without*	*that*	*he*	*still*	*knew*	*why.*	
(33)	Sie	malen,	wo	sie	sich	berühren	lassen	-	und	von	wem.
	They	*draw*	*where*	*they*	*themselves*	*touched*	*let*	-	*and*	*by*	*who*

Even though in the 'real-life' data, the comma was omitted in a lot of the cases, it is still a hint for a sentence to be a sluice when there is a comma, as in most non-sluices the comma would be incorrect.

For this purpose, the elements -5 to -3 of a sentence ([-1] is the full stop, [-2] is the wh-word) were assigned to the list `last_elements` (l.238-239). As some of the sentences are comprised of only three words or less, the exception needed to be caught (l. 240): if the sentence was not long enough, it was annotated as non-sluice.

```
236             punctuation_marks=[',', ';', '-']
237             try: #try to make a list of the last elements before the wh-word
238                 last_elements=[punctuation.tagged_hit[-5],punctuation.tagged_hit[-4],\
239                                punctuation.tagged_hit[-3]]
240             except:
241                 punctuation.row[18]='n'     #if sentence too short, predict as non-sluice
```

Figure 65: Rule 7 – `punctuation` I

The program then iterates through the punctuation marks:

```
244             #check if there's a punct mark among the last elements
245             for element in last_elements:
246                 if element[0] in punctuation_marks:
247                         punctuation.row[18]='y'     #if so, predict as sluice
```

Figure 66: Rule 7 – `punctuation` II

If one of the marks (l. 236) was present, the sentence was tagged as sluice.

```
70 false prediction(s) by punctuation:
- Wann, wann, wann.
- Ich meine, schlabbern die schon um die Knie herum, oder was.
- Tante Else eine messerscharfe Idee mitteilen, ach was.
- Aber als er auf mein geöltes Eichenparkett schoss, ganz Topagent, das hatte was.
- Ihre Maschine macht dann "ping" und sagt Ihnen: Wenn Sie dieses Foto oder jene Musik haben wollen, kostet das was.
- Ach was.

[…]
249 correct prediction(s).
```

Figure 67: Impact of `punctuation`

Again, a high number of the corpus' instances could be fetched and correctly tagged. Any wider or narrower window of last elements yielded a lower performance.

6.3.1.8 Rule 8: Check for Verbal Material (`verb_search`)

In case neither the sentence nor its 'ContextBefore' had a verb, the sentence was annotated as a non-sluice:

```
263             if verb_search.row[16]=='<none>':   #check if there is a verb in this or previous sentence
264                 verb_search.row[18]='n'     #if there's no verb, predict as non-sluice
```

Figure 68: Rule 8 – `verb_search`

For this purpose, the automatically generated annotation <none>, built in 5.2.4 was utilized. Only ten sentences were affected by this rule, but also, it only output correct results:

```
0 false prediction(s) by verb_search:
10 correct prediction(s).
```

Figure 69: Impact of `verb_search`

6.3.1.9 Rule 9: Check the Sentence Length (`sent_length`)

Investigating the sentence length in 5.2.3, it turned out that a lot of the very short sentences in the corpus were no sluices. This led to the following rule in the algorithm:

```
279             if int(sent_length.row[12])<=2:     #check if sentence is 2 words or shorter
280                 sent_length.row[18]='n'     #if so, predict as non-sluice
```

Figure 70: Rule 9 – `sent_length`

Any sentence that was two words or shorter was annotated as non-sluice.

```
4 false prediction(s) by sent_length:
- Und wo.
- Und warum.
- Und woran.
- Und was.
26 correct prediction(s).
```

Figure 71: Impact of `sent_length`

6.3.1.10 Rule 10: Check if Sentence is Negated (`negation`)

In 5.1.4, it showed that sluices are more likely to be a negated sentence than the average sentence. Therefore, the following rule was instantiated:

```
298             try: #try to make a list of the last elements before the wh-word
299                 last_elements=[negation.tagged_hit[-6],negation.tagged_hit[-5],\
300                                negation.tagged_hit[-4], negation.tagged_hit[-3]]
301             except:
302                 pass
303
304             for el in last_elements:
305                 if el[1] =='PTKNEG' : #check if one of those elements is a negation particle
306                     negation.row[18]='y' #if so, predict as sluice
```

Figure 72: Rule 10 - `negation`

`last_elements`, in this case, contains the last four words of a sentence before the wh-word. If one of them is a negation particle ('nicht', *Engl.: not*) tagged by 'PTKNEG' (Schiller, Teufel and

Stöckert 1999), it is assumed that the sentence is negated[27]. This aims especially for sluices in the 'classic' shape (see 2.4.1) where the negation particle stands very close to the wh-word. Again, any wider or narrower list of last elements yielded a lower performance.

`negation` produced 11 false positives and 121 correct predictions when put first in the algorithm:

```
11 false prediction(s) by negation:
- »Auf jeden Fall haben sich alle ganz unglaublich betrunken und ins Wohnzimmer gekotzt und weiß nicht was.
- »Das hat doch mit Größe nichts zu tun, Mensch, Pischke, da gibt's Spritzen und ich weiß nicht was.
- Mein Onkel sagt zu meinem Vater, zwei zu eins ist nicht schlecht, was.
- Und wie.
- Oder wie.
- Egal woher.
- Egal wohin.
- was wann wer warum.
- Oder warum.
- Und dann so was.
- Wirr um was.
121 correct prediction(s).
```

Figure 73: Impact of `negation`

6.3.1.11 Rule 11: The Verb 'wissen' (`wissen`)

As could be seen in 5.1.2, 'wissen' (*Engl.: to know*) is by far the most common embedding verb for sluices (57.13% of the sluices were embedded by 'wissen'). Additionally, in 5.2.4, it turned out, that more than 83% of the embedding verbs in sluices are the last verb in the sentences.

These two findings were combined in `wissen`:

```
321             if wissen.row[16]=='wissen':     #check if last verb is 'to know'
322                 wissen.row[18]='y'     #if so, predict as sluice
```

Figure 74: Rule 11 – `wissen`

If the last verbal element of the sentence was 'wissen', the sentence was predicted as sluice.

```
13 false prediction(s) by wissen:
- Dabei werden sie hoffärtig, der Statusunterschied wird zum Problem, sie fallen vom Dichter ab und halten
sich selber für Gott weiß was.
- Es gibt mehr Kneipen als Kirchen oder Galerien oder Konzerthäuser oder Clubs oder Discos oder was weiß ich
was.
- Herr Lehmann fand genug Zeit, die Flaschen in die Kühlung zu tun, die meisten Leute hatten jetzt ihr Bier,
wenn auch warmes, und die Kneipe begann sich auch schon wieder zu leeren, der Zenit war für das Einfall für
heute nacht überschritten, die Leute zogen weiter in irgendwelche anderen Kneipen und in Clubs und Discos und
was wußte Herr Lehmann was.
[...]
139 correct prediction(s).
```

Figure 75: Impact of `wissen`

`wissen` can make a lot of correct predictions. Most of the false predictions could be sieved out by the rule `regexes` later.

[27] This is a heuristic approach. Not every sentence containing a PTKNEG is negated, and 'nicht' is not the only way to negate a sentence.

6.3.1.12 Rule 12: To Be Someone (`wer_sein`)

In German, 'wer sein' (*Engl.: to be someone*) is a common figure of speech[28] to express someone's importance, sometimes in an ironic way. 'Wer' in this case, is an indefinite pronoun. While, in standard language, one would rather use 'jemand' instead of 'wer' as an indefinite pronoun, the usage of 'wer' often forms an alliteration with 'wir' (*Engl.: we*) and 'wieder' (*Engl.: again*) or 'war' (*Engl.: was*), an example that could be found in the corpora several times:

(34)	Wir	sind	wieder	wer.
	We	*are*	*again*	*someone.*
(35)	Althans	war	wer.	
	Althans	*was*	*someone.*	

As these 'wer' are always pronouns and therefore never form a sluice, `wer_sein` was developed:

```
337             if wer_sein.row[16] =='sein' and wer_sein.row[5]=='wer': #check if last verb is 'to be
338                                                                       #and wh-phrase is 'who'
339                 wer_sein.row[18]='n'     #if so, predict as non-sluice
```

Figure 76: Rule 12 - `wer_sein`

If the last verbal element was 'sein' and the wh-word was 'wer', the sentence was predicted as non-sluice.

```
0 false prediction(s) by wer_sein:
2 correct prediction(s).
```

Figure 77: Impact of `wer_sein`

While `wer_sein` has a positive, albeit not extensive impact on this dataset, it proved to be more useful on the second corpus.

6.3.1.13 Rule 13: Take Care of the Rest (`wrapup`)

`wrapup` must be the last of all pipeline steps. It is the only rule in the algorithm that cannot be moved in a different position as it tags every sentence that has not been treated yet as non-sluice:

```
354         if wrapup.row[18] != 'y' and wrapup.row[18] != 'n':
355             unlabelled+=1   #count sentences that could not be fetched by the algorithm
356             wrapup.row[18]='n'  #predict them as non-sluices
```

Figure 78: Rule 13 – `wrapup`

Only few unclassified sentences were left in the corpus which are all non-sluices:

[28] As the inflection, and therefore the shape, of 'sein' changes according to the subject, 'wer sein' cannot be handled with a Regular Expression.

```
0 false prediction(s) by wrapup:
3 correct prediction(s).
```

Figure 79: Impact of wrapup

6.3.1.14 Summary of the Rules

The 13 rules of the algorithm can be summarized in pseudocode:

For each sentence in the corpus:

1.	If a non-sluice-**regex** matches	→	non-sluice
2.	If ‘was’ or ‘welche’ have the **POS-tag ‘PIS’**	→	non-sluice
3.	If the wh-word is **wo[a-z]+ or why**	→	sluice
4.	If there is a **preposition** before the wh-word	→	sluice
5.	If the last verbal element is a **QEV**	→	sluice
6.	If the wh-word **cannot** be an **indefinite pronoun**	→	sluice
7.	If there is a **punctuation mark** in the last elements	→	sluice
8.	If there is **no verb** in Hit or ContextBefore	→	non-sluice
9.	If the sentence is **shorter** than three words	→	non-sluice
10.	If there is a **negation** particle in the last elements	→	sluice
11.	If the last verb is **‘wissen’**	→	sluice
12.	If the wh-word is **‘wer’** and the last verb is **‘sein’**	→	non-sluice
13.	**All sentences remaining**	→	non-sluice

Figure 80: Pseudocode summary of rule-based classification algorithm

6.3.2 Arranging the Rules

The performance of the model is highly dependent on the order of its rules. The impact of the rules shown above was measured when the respective rule was first in line, making sure that each rule is able to contribute to the success of the model. This was the order the rules were developed (and listed above) in:

```
377 regexes(whfinals_df)#1
378 POS(whfinals_df)#2
379 woAZ_why(whfinals_df)#3
380 prep_check(whfinals_df)#4
381 embedding(whfinals_df)#5
382 not_pronoun(whfinals_df)#6
383 punctuation(whfinals_df)#7
384 verb_search(whfinals_df)#8
385 sent_length(whfinals_df)#9
386 negation(whfinals_df)#10
387 wissen(whfinals_df)#11
388 wer_sein(whfinals_df)#12
```

Figure 81: Original order of rules

This is not necessarily the ideal order: In this case, `woAZ_why` (l. 386) assumes that the sentence 'Warum, warum.' (*Engl.: Why, why.*) is a sluice because almost all other sentences with 'warum' are sluices. In this case, however, this is incorrect. 'Warum, warum.' is not a sluice.

```
5 false prediction(s) by woAZ_why:
- was wann wer warum.
- Die Depression hat ausgedient, die Hysterie fliegt wieder aus, wer wo wann, mit wem, was, um Gottes willen, warum.
- Oder warum.
- Warum, warum.
- Ich will anerkannt werden als einer, der ihnen zeigt, warum und wofür und wohin.
169 correct prediction(s).
```

Figure 82: False predictions by woAZ_why

This could have been avoided if, for instance, the function `sent_length` had been put higher up in the algorithm than `woAZ_why`: it would have classified 'Warum, warum.' as non-sluice because it is so short. But, put higher up in the rank, `sent_length` makes a lot more false classifications than `woAZ_why`.

All the rules interfere with each other in this way. Again, precision, recall, and F1-Score were calculated. This way, the different arrangements of rules could be compared. After shuffling the rules and testing as many arrangements as possible, the following organization yielded the best result:

```
377 regexes(whfinals_df)#1
378 wer_sein(whfinals_df)#12
379 prep_check(whfinals_df)#4
380 verb_search(whfinals_df)#8
381 negation(whfinals_df)#10
382 POS(whfinals_df)#2
383 woAZ_why(whfinals_df)#3
384 punctuation(whfinals_df)#7
385 not_pronoun(whfinals_df)#6
386 sent_length(whfinals_df)#9
387 embedding(whfinals_df)#5
388 wissen(whfinals_df)#11
```

Figure 83: Most successful organization of rules

6.3.3 Model Evaluation – Seen Data (Wh-Finals Corpus)

To be able to compare the performance of the rule-based approach with the performance of the machine learning model, the same performance measures were implemented.

6.3.3.1 Heatmap and False Classifications

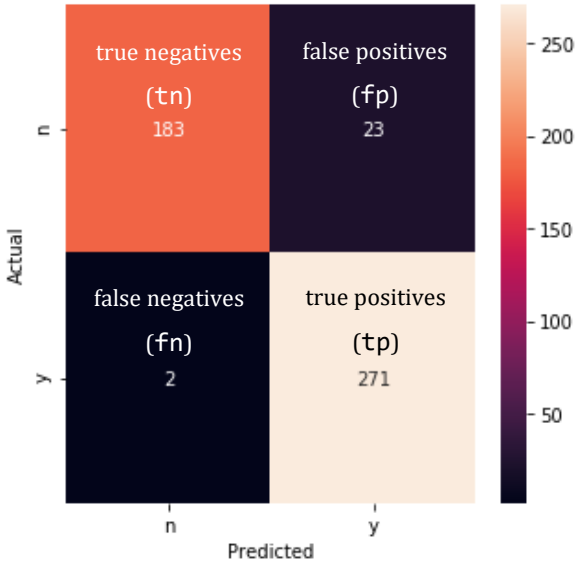


Figure 84: Heatmap on predictions (rule-based approach, wh-finals corpus)

Again, this heatmap shows that there are much less false classifications (bottom left and top right square) than correct classifications; recognizable by the darker colours and the lower numbers. The model produced 23 false positives (sentences were tagged as sluices, even though they were none) and 2 false negatives (actual sluices were not tagged as such).

Equivalent to the ML algorithm, the 2+23 falsely tagged sentences were output, too:

```
Classified as sluice, but actually a non-sluice (false positives): 23

- Ruhm ohne Titelblätter, Umfragen, Anfragen, Nachfragen, Geburtsjahr, Ort, Platz, Lieblingsfarbe, Lieblingsblume,
Lieblingsessen, Sport, Hobby, Ferien wo und wann.
- Ich meine, schlabbern die schon um die Knie herum, oder was.
- Befreiung von was.
- »Auf jeden Fall haben sich alle ganz unglaublich betrunken und ins Wohnzimmer gekotzt und weiß nicht was.
- »Das hat doch mit Größe nichts zu tun, Mensch, Pischke, da gibt's Spritzen und ich weiß nicht was.
- Mein Onkel sagt zu meinem Vater, zwei zu eins ist nicht schlecht, was.
- - Zum Aufheben für wen.
- Wenn es einen Musiker vorzustellen gilt, schreiben Kritiker gern, der erinnere an, der klinge wie.
- Aber wie.
- Oder wie.
- Der Schatz meiner Mutter, der ich bin, ist schon beinahe aufgegessen, und es gibt keinen Nachschlag, oder wie.
- Denn da ist keiner, aber auch nicht einer ohne Schnauz und Relativsätze mit: die, wo.
- was wann wer warum.
- Die Depression hat ausgedient, die Hysterie fliegt wieder aus, wer wo wann, mit wem, was, um Gottes willen, warum.
- Warum, warum.
- Auf Veranstaltungen ans Mikrofon geschubst werden, mit dem Spruch: "Nun geh doch nach vorne und sag was.
- Wirr um was.
- Nächster Vorschlag: (ich hatte ein Exemplar der Glienicker Kinderzeitung dabei) Zeitung machen - Überlegungen ob
und wie.
- Ich meine ja dieses eine einzige Wort: wie.
- Pocht mal wieder, wie.
- Wer bereitet uns vor auf die Überquerung, Überquerung von wo zu wo.
- Ich will anerkannt werden als einer, der ihnen zeigt, warum und wofür und wohin.
- Niemand kann alles wissen, jeder muss Wissen in Maßen durch Vertrauen ersetzen, gerade in der Wissensgesellschaft,
aber: Trau, schau, wem.


Classified as non-sluice, but actually a sluice (false negatives): 2

- Ob nun Schwarm, Ableger oder Volk: Kaufen Sie niemals Bienen ohne die Seuchenfreiheitsbescheinigung, die Ihnen
gesunde Völker garantiert und die außerdem vorgeschrieben ist, wenn Sie Ihre Bienen beziehen, gleichviel woher.
- »... und ich sage dir gleich, warum.
```

Figure 85: Falsely predicted documents (rule-based approach, wh-finals corpus)

6.3.3.2 Precision, Recall, and F1-Score

Only the measures of precision, recall and F1-Score show the success of the model:

```
Precision:  0.9218
Recall:  0.9927
F1-Score:  0.9559
```

Figure 86: Precision, recall and F1-Score (rule-based approach, wh-finals corpus)

The overall F1-Score of the model is very high. The extraordinarily high recall is the result of only two false negatives – which might also be a sign for how strongly the model was tailored towards this specific dataset.

6.3.4 Model Evaluation – Unseen Data (Second Corpus)

Even though the rules of the algorithm were developed as universally as possible, the result would still be influenced by the data it was developed on. The RegEx for frozen expressions, for example, caught those frozen expressions that were present and seen in the wh-finals corpus.

In order to be able to assess the algorithm's performance more independently from the initial data, the second corpus (194 sentences from the 'Berliner Zeitung' (*Berlin Newspaper*), see

4.4.1) was built and annotated (complying with the standards described in 4.) and processed by the pipeline. Again, the rules were shuffled until no higher performance could be obtained.

```
377 regexes(whfinals_df)#1
378 POS(whfinals_df)#2
379 wer_sein(whfinals_df)#12
380 prep_check(whfinals_df)#4
381 wissen(whfinals_df)#11
382 verb_search(whfinals_df)#8
383 sent_length(whfinals_df)#9
384 not_pronoun(whfinals_df)#6
385 woAZ_why(whfinals_df)#3
386 embedding(whfinals_df)#5
387 punctuation(whfinals_df)#7
388 negation(whfinals_df)#10
```

Figure 87: Most successful organization of rules for second corpus

6.3.4.1 Heatmap and False Classifications

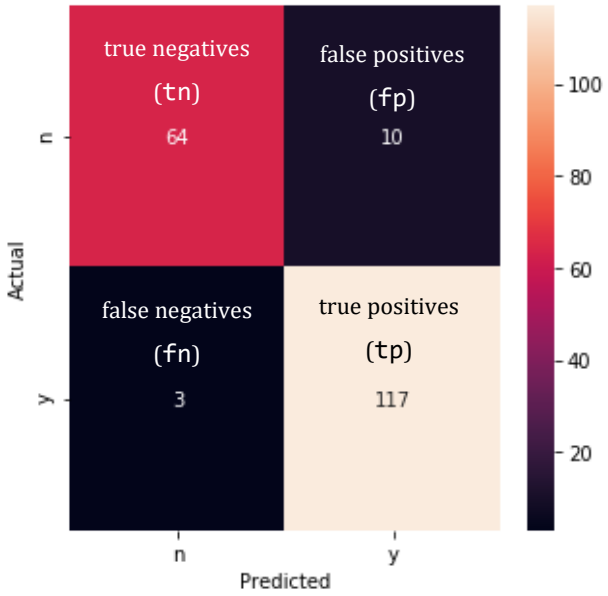


Figure 88: Heatmap on predictions (rule-based approach, second corpus)

In this case, too, the number of correct predictions is significantly higher than the number of false predictions. Those are the 10+3 wrong classifications:

```
Classified as sluice, but actually a non-sluice (false positives): 10

- Eine Erbschaftsangelegenheit bildet den Ausgangspunkt, und alsbald erzählen so viele anwesende Personen von so
vielen nicht anwesenden Personen, dass kaum einer noch begreift, wer mit wem und weshalb.
- Klar doch, so langsam verlieren die Herren auf St. Pauli die Übersicht, wer hier was und wieso.
- Sie dampfte und tutete, weiß der Kuckuck woher.
- Auch das nur ein Wort, aufgetaucht von weiß ich woher.
- Von Berlinern, die nicht wissen wohin, zu Berlinern, die nicht wissen woher.
- Aus der Tiefe des "diskursiven Raumes" oder sonst woher.
- Gusselnow, der die Sonne Aserbaidschans schmerzlich vermißt, weiß nicht wohin.
- Ein Skandal sei es beispielsweise, daß das Landesmuseum für Ur- und Frühgeschichte immer noch nicht wüßte wohin.
- Frau Goschev setzt sich in den Aufenthaltsraum mit den hellgebeizten Holztischen und den Kruzifixen an der Wand und
weint, weil sie nicht weiß, wohin.
- Wer, warum und mit wem.


Classified as non-sluice, but actually a sluice (false negatives): 3

- Nur: Sie sind uneinig, wer.
- Nur wann, ist die große Frage, und wie.
- ---- 5. Februar Immer wieder begegnen einem diese Leute, die jemandem ähnlich sehen, den man von Film oder
Fernsehen kennt, oft weiß man gleich, wem.
```

Figure 89: Falsely predicted documents (rule-based approach, second corpus)

Also, the difference in colour in the upper left and lower right quadrant of the heatmap shows, how this corpus is not as well balanced as the wh-finals corpus[29] (see Figure 13). The difference in colour could also hint towards a great difference between precision and recall, which is not the case as can be seen next.

6.3.4.1 Precision, Recall, and F1-Score

```
Precision:  0.9213
Recall:  0.975
F1-Score:  0.9474
```

Figure 90: Precision, recall and F1-Score (rule-based approach, second corpus)

As expected, the performance of the model is not as high any more once the data is unknown to the developer, but the algorithm still performs satisfactorily. Both recall and precision dropped slightly, leading to an overall F1-Score of 94.74%.

6.4 Comparing the Models

The following table compares the performance and characteristics of the ML model and the rule-based model for sluice detection.

[29] While this could be a problem for an ML algorithm, the performance of the rule-based algorithm is not influenced by the balance of the data.

	ML approach		Rule-based approach	
	wh-finals corpus (original corpus)	third corpus (larger corpus)	wh-finals corpus (seen data)	second corpus (unseen data)
1. Precision	.86	.8647	.9218	.9213
2. Recall	.9773	.9664	.9927	.975
3. F1-Score	.9149	.9127	.9559	.9474
4. Universality	High		Low	
5. Transparency of the algorithm	Black box		Highly transparent	
6. Changes and improvement of the algorithm	Uneducated guessing		Controlled changes	
7. Linguistic knowledge for development	No knowledge needed		Some knowledge needed	
8. Preparational workload	Low		High	
9. Need for annotated data running	Annotation needed		No annotation needed	
10. Amount of data needed	High amount needed		Amount is irrelevant	

Table 5: Comparison of the two models

Considering the performance measures precision, recall and F1-Score (1-3), the rule-based approach performs remarkably better than the ML algorithm.

The universality (4), meaning the applicability of the code onto different datasets or even entirely different linguistic problems, is a lot higher in the ML algorithm. The ML code can take

any annotated textual input that can be approached with a TF-IDF document classifier and perform its action on it. The rule-based approach, however, is less robust: The input data must have the same shape as the corpora used in this task. Also, it was created to detect sluices – nothing else.

A lot of ML algorithms are 'black boxes' (5):

> '[...] Most [...] machine learning approaches are, essentially black boxes, in which you can't really inspect how the algorithm is accomplishing what it is accomplishing.' (Russell 2016)

Li's (2018) algorithm is no exception: Although it outputs what classifications were done wrong (see e.g. Figure 40), one cannot find out why they were done wrong. As Russell (2016) says, it is not traceable why the document classifier does what it does. An example of this is the output of false classifications (see 6.2.5): Only two out of 18 of the false classifications in the first corpus could be found in the false classifications of the third corpus, even though the third corpus was only an extension of the first. There is no explanation why the sentences that were classified falsely differed from the first to the third corpus.

The actions of the rule-based algorithm, on the contrary, are easily comprehensible. For example, if a sentence is shorter than three words and the rule `sent_length` is first in the algorithm, the document will be categorized as non-sluice. If another step of the algorithm was performed before `sent_length`, e.g. the rule `POS` found that this sentences wh-word is 'wer' and has the POS-tag 'PIS', it will be classified as sluice – and so on. Knowing the rules of the algorithm, it is always observable why the algorithm classifies a sentence the way it does.

As a result, improvements (6) in the ML code can only be done 'blindly' – like changing the `min_df` from 5 to 3 (see 6.2.2.2) or changing the `ngram_range` (see Figure 91). Even if the developer knows what these features mean, it is not traceable what altering them entails. Higher results must be obtained by trial and error.

```
45 #initialize tfidf vectorizer
46 tfidf = TfidfVectorizer(sublinear_tf=True, min_df=3, norm='l2', encoding='utf-8', ngram_range=(1, 2))
```

Figure 91: 'Blind' improvements in the ML code

Enhancing the ML code's performance therefore is uneducated guesswork, while the rule-based algorithm can be changed in a controlled way, considering the data: Does it look like the most 3-word-sentences in the corpus are non-sluices? Then line 279 would be changed to `<=3`.

```
279             if int(sent_length.row[12])<=2:
280                 sent_length.row[18]='n'
```

Figure 92: Easy changing of rules

For the development of the ML approach no linguistic knowledge is needed (7) as the task of distinguishing between sluice and non-sluice is approached as a data science problem, not a linguistic problem. The only linguistically inspired part of the algorithm is the application of the TF-IDF vectorizer, which is a built-in function of scikit-learn and only requires a small number of parameters (like, e.g., the n-gram range) (scikit-learn 0.20.2 documentation 2018). Developing the rule-based algorithm did require some linguistic expertise in the fields of syntax (e.g. finding the embedding verb), morphology (e.g. interpreting and applying the results from the POS-tagger), semantics and NLP (e.g. using RegEx and TreeTagger).

Judging the preparational workload (8) of the ML algorithm is difficult as it was adopted from Li (2018). Still, the code is, as opposed to the rule-based approach, not tailored to the specific problem of distinguishing between sluices and non-sluices. It is a generic pipeline to classify documents into predetermined classes. Therefore, there was no linguistic research conducted prior to the creation of the algorithm, which was one of the most time-consuming parts in the development of the rule-based algorithm. Furthermore, the ML algorithm utilizes a lot of built-in libraries, functions and methods. While it does require a lot of Python coding skill to arrange the parts of the algorithm correctly, the actual task of classifying the documents is done by a prefabricated function. It is reasonable to assume that the ML algorithm took less time to develop than the rule-based algorithm.

Since the rule-based algorithm now is set up and validated, it does not need annotated data anymore (9). Any wh-final sentence could be fed into the algorithm and processed by it, determining if it's a sluice or not. The ML algorithm, in order to be able to deliver valid and profound results, needs a large amount (10) of annotated input sentences. This is not the case for the rule-based approach: it could get one unannotated sentence as input and deliver a result with the same confidence as if the input was 2000 sentences.

7. Discussion and Future Work

7.1 Wh-Finals Corpus

Corpus annotations are usually done by more than one domain expert to level out any biases or misconceptions the annotator might have. An annotation done by several annotators typically has a higher quality and is more reliable. Hence, in order to secure the wh-finals corpus' quality and validity, its annotation should be redone by more domain experts.

7.2 Rule-Based Algorithm

The wh-finals corpus is a data resource of sentences with a final wh-word. It only contains sentence-final sluices and non-sluices that end in a wh-word and a period (WH+PERIOD). Other patterns like WH+EXCLAMATIONMARK or sentence-internal sluices are not in the corpus. As the algorithm is built based on this corpus, sentence-internal sluices like (11a) cannot be processed by the algorithm (Anand and McCloskey 2015a: 1):

(11a)	Es	ist	klar,	dass	sich	die	Uni	ändern	muss,	aber	wie	[ ]	ist	weniger	klar.
	It	*is*	*clear,*	*that*	*itself*	*the*	*uni*	*change*	*must,*	*but*	*how*	*[ ]*	*is*	*less*	*clear.*

The rule-based algorithm is a distinguishing algorithm between wh-final sentences that are sluices and those that are not. An excellent advancement of the algorithm would be the ability to find sluices in plain text, without being limited to wh-final sentences and therefore sentence-final sluices.

Arranging the rules so that the algorithm yields the highest possible accuracy at the end of the rule-based algorithm is done manually so far. As a developer, it was possible to obtain an intuition on when the performance would reach its highest point. But with 12 interchangeable rules, the number of possible arrangements is 12 factorial:

$$12! = 12 * 11 * 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 479.001.600$$

There are more than 479 million ways to arrange the rules. To make sure that the best solution is found, the rule-based algorithm should be augmented by a rule-shuffling function that automatically finds out the most performant order of rules. At the same time, if this function tried every possible combination of rules, it would be extremely computationally expensive. Here, a middle way needs to be found.

7.3 ML Algorithm

TF-IDF is one of the most popular approaches (Xu *et al.* 2013: 2) to represent text features, but it might not be the best way to represent the wh-final sentences: 'Especially when labelled data

is limited […], or the text documents are short […], many features are rarely observed within the training corpus' (Xu *et al.* 2013: 2).

An 'alternative weighting function to TF-IDF' (Jurafsky and Martin 2018: 116) is, e.g., PPMI (positive pointwise mutual information). PPMI is 'a measure of how often two [words x and y] occur, compared with what we would expect if they were independent' (Jurafsky and Martin 2018: 116).

As the wh-final sentences do not share a mutual topic, even a simple bag-of-words approach counting word occurrences could yield good results. Scikit-learn offers the method `CountVectorizer` for this purpose: it returns an encoded vector with a length of the entire vocabulary of all documents and an integer count for the number of times each word appeared in the document (Brownlee 2017).

Another alternate vectorizer built in scikit-learn is `HashingVectorizer`. Especially once the dataset grows larger, `HashingVectorizer` can be used to reduce the amount of memory required (scikit-learn 0.20.2 documentation 2018). Instead of storing the words as strings, `HashingVectorizer` transforms them into integers which is much more memory efficient. This process, however, cannot be undone (Brownlee 2017).

Additionally, it would be interesting to find out why the ML algorithm performed worse on the slightly bigger dataset (see 6.2.5) than on the original datset. It was still balanced enough, and the additional data was pulled from a similar corpus as the original data. There seem to be more influences on the performance of the algorithm than previously assumed.

7.4 Extension of the Algorithm: Sluice Resolution

Detecting the sluices is the first step in a smooth computational sluice processing. As Nielsen (2005) pointed out, ellipsis resolution consists of three parts (Nielsen 2005: 28):

1. Detecting ellipsis occurrences
2. Identifying antecedents (in sluicing: sponsor clause)
3. Resolving the ellipsis

The antecedent of a VPE (which is the phenomenon Nielsen (2005) dealt with) is the material that got elided, i.e. needs to be rebuilt:

(1) John **likes candy**, but Bill doesn't [like candy].

In the case of sluices, the rebuilt material is the sponsor clause. The next stage in the process would therefore be the recognition of a sluice's sponsor clause.

In the simplest case, to resolve the sluice, the sponsor clause ('wir sühnten' in (36)) has to be inserted after the wh-remnant, either in the same order (36) or shuffling the constituents[30] (37):

Legend: Subject Verbal complex Object Antecedent

	PRESLUICE/SPONSOR	EMBEDDING CLAUSE	REMNANT	REBUILT MATERIAL
(36)	Wir sühnten *We expiated,*	wiewohl wir nicht wussten, *even though we didn't know,*	was *what*	[wir sühnten]. *[we expiated].*
(37)	Er kam ihr bekannt vor, *He seemed familiar to her,*	doch sie konnte sich nicht erinnern, *but she could not remember,*	woher *from where*	[er ihr bekannt vorkam]. *he seemed familiar.*
(38)	Sein Herz war anderswo, *His heart was somewhere else,*	doch durfte keine der Geliebten ahnen, *but none of his mistresses could know,*	wo *where*	[sein Herz war]. *[his heart was].*

As can be seen in (38), the antecedent 'anderswo' (*Engl.: somewhere else*, adverbial phrase) does not get rebuilt into the gap, it is therefore not part of the sponsor clause.

Sluice resolution is by far not always this easy: the sponsor clause could be in the preceding sentence or not available at all:

	CONTEXT BEFORE	HIT
(39)	FISCHER: Natürlich! *Fisher: Of course!*	Ich kann Ihnen auch sagen, warum. *I can also tell you, why.*

Although (39) is clearly a sluice, it cannot be resolved with only one sentence of context.

Even if the sponsor clause is available, the resolution can turn quite complex, especially because of the verbal bracket and the 'word order variability in German clauses [that] is considerably greater than [in] […] English' (Kempen and Harbusch 2017: 1).

	PRESLUICE	EMBEDDING CLAUSE	REMNANT	REBUILT MATERIAL
(40)	Keine der Gestalten auf dem Gemälde ‚Der Frühling in den Uffizien' denkt an den Frühling, keine freut sich, ich denke	- der Künstler wusste,	warum	[keine der Gestalten auf dem Gemälde 'Der Frühling in den Uffizien' an den Frühling denkt, keine sich freut].
	None of the figures in the painting 'Der Frühling in den Uffizien' thinks of spring, none is happy, I think	*- the artist knew*	*why*	*[none of the figures in the painting 'Der Frühling in den Uffizien' thinks of spring, none is happy.]*

Not only there is additional material ('ich denke') in the presluice that is neither part of the sponsor clause nor the antecedent. There are several variations of the rebuilt material that are grammatical and feasible as it is not entirely clear what the sluiced material points towards:

[30] The verbal bracket can be a reason for a mandatory shuffling of constituents in German.

- …keine der Gestalten […] an den Frühling denkt, keine sich freut.
- …keine der Gestalten […] an den Frühling denkt, keine der Gestalten […] sich freut.
- …keine der Gestalten […] an den Frühling denkt.
- …keine der Gestalten […] sich freut.

McShane and Babkin‘s (2016) observation, that ‘[in VPE,] syntactically simple elliptical structures are easier to resolve than syntactically complex ones’ (McShane and Babkin 2016: 3), applies to sluices, too. In a simple sluice like (36), the sponsor can easily be found and rebuilt: It is the first noun/pronoun and verb of the sentence, or, if a parser could be involved in the process, the subject and predicate of the main clause. As soon as the structure is more complex though, the resolution of the sluice will be more difficult than that.

It turns out that there are still some tasks to master on the way to successful automatic sluice processing.

8. CONCLUSION

This thesis is a contribution to automatically detecting sluices. The literature review and the theoretical examination of the phenomenon resulted in a broad understanding of what, in theory, sluicing means and ultimately inspired a heuristic approach to sluice detection: pulling wh-final sentences from a large corpus. The creation of the wh-finals corpus helped to gain an insight to the phenomenon of sluicing in 'real life' – how exactly and in what contexts is this type of ellipsis used by actual speakers? Which constructions look like sluices at the first glance but turn out not to be an ellipsis? The high number of non-sluices in the wh-finals corpus (43%) proved that a simple RegEx query on a corpus is not enough to detect sluices. The insights obtained by manually and automatically inspecting the wh-finals data helped developing a set of 13 rules to distinguish between a sluice and a wh-final sentence that is no sluice.

The execution of these rules is computationally inexpensive and does not require any external software other than the TreeTagger (Schmid 2018). Every rule has the same architecture: If a certain statement is true, the sentence will be classified as a sluice or non-sluice, depending on the rule. The statements compare, e.g., the POS of the wh-phrase or the verbal material, look for specific figures of speech or check the length of the sentence. If the sentence, for example, has two words or less, it will be classified as a non-sluice. If there is a punctuation mark in the last elements, it will be predicted as a sluice. Every rule proved to contribute to the success of the model, even though they were all generalisations. Of course, not every single two-word-sentence is a non-sluice and not every single sentence with a punctuation mark in the last elements is a sluice. However, the rules of the algorithm are able to capture the general tendency eminently well: On unseen data, the model yielded an F1-Score of 94.74%.

The same task – deciding if a sentence is a sluice or not – was completed by an ML document classification algorithm (Li 2018). It applied the TF-IDF model onto the dataset and used a Linear Support Vector Machine to classify the sentences. This supervised document classification pipeline obtained an F1-Score of 91.49% on the wh-finals corpus.

While the F1-Score is an unequivocal proof that the rule-based algorithm performs better than the ML algorithm in detecting sluices, other characteristics of the algorithms and their development must be considered, too: the time and knowledge resources required to set up the code, the application of the algorithm onto the data, its applicability to different datasets and research problems – in these domains the ML algorithm clearly outperforms the rule-based algorithm.

For deciding which algorithm is generally better, the specific use case needs to be considered: If, on the one hand, a particularly high F1-Score is desired, the rule-based approach is the better choice. If, on the other hand, the algorithm needs to be implemented quickly and reused for

other purposes later, the ML approach would work best. However, the major advantage of the rule-based algorithm in addition to its high performance is its ability to be executed without initial data input, as sparse (annotated) data is a common problem in NLP, especially in languages other than English.

Future investigations of German sluices will require a larger corpus of the phenomenon. The developed rule-based algorithm can help to automatically build such a resource by reliably distinguishing between actual sluices and non-elliptical wh-final sentences, even without initial annotated data.

Furthermore, the extraordinarily high performance of the algorithm in detecting sluices makes it a reliable first building block of a sluice resolution pipeline. The algorithm is therefore a valuable contribution to the successful automatic computational processing of sluices.

List of Figures

List of Tables

Annex

- Folder 'Code_Sluice_or_Nonsluice':
  - `README.txt`: explanation on how to execute the code
  - `precrec.py`: script for calculating precision, recall, and F1-score of rule-based algorithm
  - `process_stopwords.py`: script for turning document with stop words into python list
  - `Step0_POStag.py`: script for POS-tagging 'Hit' and 'ContextBefore' (requires TreeTagger and Ubuntu, but does not need to be executed as tagged corpora are provided)
  - `Step1_computational_annotation.py`: annotates corpus as described in 5.2; input: `first_corpus_POS_tagged.csv` or `second_corpus_POS_tagged.csv`
  - `Step2_data_investigation.py`: investigates wh-finals data as described in 5.
  - `Step3_ML_doc_class.py`: machine learning document classification algorithm, input: `first_corpus_POStagged.csv` or `third_corpus.csv`
  - `Step3_rule_based_doc_class.py`: rule-based document classification algorithm, input: `whfinals_corpus.csv`
  - Folder '`corpora`':
    - `first_corpus_POStagged.csv` (fully tagged and manually annotated corpus)
    - `second_corpus_POStagged.csv` (fully tagged, only annotated with sluice/non-sluice)
    - `stopwords.txt` (German stop words)
    - `third_corpus_POStagged.csv` (fully tagged, only annotated with sluice/non-sluice)
  - Folder '`output`' (empty):
    - Will contain `whfinals_corpus.csv` after execution of `Step1_computational_annotation.py`

References

Aelbrecht, Lobke. 2009. “You have the right to remain silent. The syntactic licensing of ellipsis”. Dissertation, Katholieke Universiteit.

Algryani, Ali. 2011-2012. “Sluicing in Libyan Arabic”. Al-'Arabiyya: Georgetown University, 41–63.

Anand, Pranav, and Daniel Hardt. 2016. “Antecedent Selection for Sluicing: Structure and Content”. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas, 1234–43.

Anand, Pranav, and James McCloskey. 2014. “Sluicing Annotation Guide”.

Anand, Pranav, and James McCloskey. 2015a. “Annotating the Implicit Content of Sluices”.

Anand, Pranav and James McCloskey. 2015b. “Santa Cruz Ellipsis Project” <http://ohlone.ucsc.edu/SCEC/>. (10 Jan. 2019).

Bansal, Shivam. 2018. “A Comprehensive Guide to Understand and Implement Text Classification in Python” <https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/>. (26 Feb. 2019).

Barros, Matthew. 2014. “Sluicing and Identity in Ellipsis”. Dissertation, Rutgers, The State University of New Jersey.

Bock, Tim. 2018. “What is Feature Engineering?” <https://www.displayr.com/what-is-feature-engineering/>. (26 Feb. 2019).

Bos, Johan, and Jennifer Spenader. 2011. “An annotated corpus for the analysis of VP ellipsis”. *Language Resources and Evaluation* 45: 463–94.

Bronshtein, Adi. 2017. “Train/Test Split and Cross Validation in Python” <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6>. (26 Feb. 2019).

Brownlee, Jason. 2014. “Discover Feature Engineering, How to Engineer Features and How to Get Good at It” <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>. (26 Feb. 2019).

Brownlee, Jason. 2017. “How to Prepare Text Data for Machine Learning with scikit-learn” <https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/>. (26 Feb. 2019).

Castle, Nikki. 2017. “Supervised vs. Unsupervised Machine Learning” <https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms>. (26 Feb. 2019).

Chung, Sandra. 2005. “Sluicing and the Lexicon: The Point of No Return”. In Rebecca T. Cover and Yuni Kim, eds. BLS 31: General Session and Parasession on Prosodic Variation and Change, 73–91.

Chung, Sandra. 2013. “Syntactic Identity in Sluicing: How Much and Why”. Linguistic Inquiry. Vol. 44, 1–44.

Chung, Sandra, William Ladusaw, and James McCloskey. 2011. “Sluicing. Between Structure and Inference”.

Cirić, Ivan. 2018. “Intro to evaluation metrics for predictive models and how to use them in Spark MLlib” <http://www.multicom.hr/evaluation-metrics-for-predictive-models-and-how-to-use-them-in-spark-mllib/>. (26 Feb. 2019).

Condliffe, Jamie. 2018. “The Case Against Deep-Learning Hype” <https://www.technologyreview.com/the-download/609875/the-case-against-deep-learning-hype/>. (26 Feb. 2019).

Dayal, Veneta, and Roger Schwarzschild. 2010. “Definite Inner Antecedents and Wh-Correlates in Sluices”. In Peter Staroverov, Daniel Altshuler, Aaron Braver, Carlos A. Fasola and Sarah Murray, eds. Rutgers Working Papers in Linguistics. Vol. 3, 92–114.

Diaz, Gene. 2016. “stopwords-de/stopwords-de.txt at master” <https://github.com/stopwords-iso/stopwords-de/blob/master/stopwords-de.txt>. (26 Feb. 2019).

Dorash, Maryna. 2017. “Machine Learning vs. Rule Based Systems in NLP” <https://friendly-data.io/blog/machine-learning-vs-rule-based-systems>. (26 Feb. 2019).

Freed, Andrew R. 2017. “Demo of natural language processing with rules and machine-learning based approaches” <https://freedville.com/blog/2017/01/13/demo-of-natural-language-processing-with-rules-and-machine-learning-based-approaches/>. (26 Feb. 2019).

Fromkin, Victoria A., Susan Curtiss, Bruce P. Hayes, Nina Hyams, Keating, Patricia A., Koopman, Hilda, Pamela Munro, Dominique Sportiche, Edward P. Stabler, Donca Steriade, Tim Stowell, and Anna Szabolcsi, eds. 2000. *Linguistics. An Introduction to Linguistic Theory.* Malden, MA: Blackwell Publishers.

GECCo Project. 2017. “Overview on the Annotation of Ellipses in English and German Corpus Texts (GECCo-Corpus) with MMAX2”.

George, Benjamin Ross. 2011. “Question Embedding and the Semantics of Answers”. Dissertation, University of California.

Hui, Jonathan. 2018. “mAP (mean Average Precision) for Object Detection” <https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173>. (26 Feb. 2019).

Jurafsky, Daniel, and James H. Martin. 2018. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*.

Kachkach, Ahmed. 2018. “Problem-solving with ML: automatic document classification” <https://cloud.google.com/blog/products/gcp/problem-solving-with-ml-automatic-document-classification>. (26 Feb. 2019).

Karttunen, Lauri. 1977. “Syntax and semantics of questions”. Linguistics and Philosophy. Vol. 1, 3–44.

Kateb, Faris, and Jugal Kalita. 2015. “Classifying Short Text in Social Media: Twitter as Case Study”. *International Journal of Computer Applications*.

Kempen, Gerard, and Karin Harbusch. 2017. “How flexible is constituent order in the midfield of German subordinate clauses? A corpus study revealing unexpected rigidity”.

Klein, Wolfgang et. al. 2018. “DWDS. Das Wortauskunftssystem zur deutschen Sprache in Geschichte und Gegenwart” <https://www.dwds.de/>.

Kumar, Ajitesh. 2018. “Dummies Notes - Supervised vs Unsupervised Learning” <https://vitalflux.com/dummies-notes-supervised-vs-unsupervised-learning/>. (26 Feb. 2019).

Lahiri, Utpal. 2002. “Questions and Answers in Embedded Contexts”. *Oxford Studies in Theoretical Linguistics*.

Li, Susan. 2018. “Multi-Class Text Classification with Scikit-Learn” <https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f>. (26 Feb. 2019).

Marr, Bernard. 2016. “The Top 10 AI And Machine Learning Use Cases Everyone Should Know About” <https://www.forbes.com/sites/bernardmarr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/#7e3c74f894c9>. (26 Feb. 2019).

McShane, Marjorie, and Petr Babkin. 2015. “Automatic Ellipsis Resolution: Recovering Covert Information from Text”. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. Austin, Texas: AAAI Press.

McShane, Marjorie, and Petr Babkin. 2016. “Detection and Resolution of Verb Phrase Ellipsis”. Linguistic Issues in Language Technology - LiLT. Vol. 13, 1–34.

Merchant, Jason. 2001. *The Syntax of Silence. Sluicing, Islands, and the Theory of Ellipsis*.

Merchant, Jason. 2003. *Sluicing:* SynCom Case 98.

Merchant, Jason. 2012. “Ellipsis”. In Tibor Kiss and Artemis Alexadiou, eds. Syntax: An international handbook of contemporary syntactic research. Berlin: Walter de Gruyter.

Nicholson, Chris V. 2018. “Skymind Enterprise AI Wiki. Bag of Words & TF-IDF” <https://skymind.ai/wiki/bagofwords-tf-idf>. (26 Feb. 2019).

Nicklason, Lars. 2018. “How much can a computer understand?” <https://www.chalmers.se/en/areas-of-advance/ict/news/Pages/How-much-can-a-computer-understand.aspx>. (26 Feb. 2019).

Nielsen, Leif Arda. 2005. “A corpus-based study of Verb Phrase Ellipsis Identification and Resolution”. Dissertation, King's College.

Nooney, Kartik. 2018. “Deep dive into multi-label classification. Toxic-comments classification” <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>. (26 Feb. 2019).

Otto, Mirko. 2018. *A Python module for interfacing with the TreeTagger by Helmut Schmid*.

pandas 0.23.4 documentation. 2018 <https://pandas.pydata.org/pandas-docs/stable/index.html>. (26 Feb. 2019).

Parkes, Duncan. 2017. "Machine Learning vs. Rules Systems" <https://deparkes.co.uk/2017/11/24/machine-learning-vs-rules-systems/>. (26 Feb. 2019).

Perez, Sarah. 2018. "Twitter's doubling of character count from 140 to 280 had little impact on length of tweets" <https://techcrunch.com/2018/10/30/twitters-doubling-of-character-count-from-140-to-280-had-little-impact-on-length-of-tweets/>. (26 Feb. 2019).

Roelofsen, Floris, Nadine Theiler and Maria Aloni. 2017. "A semantic account of the selectional restrictions of some (anti-)rogative verbs" <https://nadinetheiler.net/papers/SALT2017_TheilerRoelofsenAloni_slides.pdf>. (26 Feb. 2018).

Rosén, Victoria, Koenraad de Smedt, Paul Meurer, and Helge Dyvik. 2012. "An open infrastructure for advanced treebanking". In Jan Hajič, Koenraad de Smedt, Marko Tadić and António Branco, eds. META-RESEARCH Workshop on Advanced Treebanking at LREC2012. Istanbul, Turkey, 22–29.

Ross, John Robert. 1969. "Guess Who?". In Robert Binnick, Alice Davison and Georgia Green, eds. Papers from the 5th regional meeting of the Chicago Linguistic Society. Chicago, Illinois.

Russell, Stuart. 2016. "The Dawn of Artificial Intelligence". https://samharris.org/podcasts/the-dawn-of-artificial-intelligence1/.

Sandhu, Gurpal. 2018. "TF-IDF vs Bag-of-Words" <https://www.kaggle.com/occam19/tf-idf-vs-bag-of-words>. (26 Feb. 2019).

Sarkar, Dipamjam. 2018. "Understanding Feature Engineering (Part 3). Traditional Methods for Text Data" <https://towardsdatascience.com/understanding-feature-engineering-part-3-traditional-methods-for-text-data-f6f7d70acd41>. (26 Feb. 2019).

Schiller, Anne, Simone Teufel, and Christine Stöckert. 1999. *Guidelines für das Tagging deutscher Textcorpora mit STTS. Kleines und großes Tagset*.

Schmid, Helmut. 1994. *Probabilistic Part-of-Speech Tagging Using Decision Trees*.

Schmid, Helmut. 1995. "Improvements in Part-of-Speech Tagging with an Application to German" <http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>. (26 Feb. 2019).

Schmid, Helmut. 2018. *TreeTagger. A language independent part-of-speech tagger*.

scikit-learn 0.20.2 documentation. 2018 <https://scikit-learn.org/stable/documentation.html>. (26 Feb. 2019).

Sebastiani, Fabrizio. 2002. "Machine Learning in Automated Text Categorization". ACM Computing Surveys. Vol. 34, 1–47.

Shung, Koo Ping. 2018. "Accuracy, Precision, Recall or F1?" <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>. (26 Feb. 2019).

Somers, James. 2017. "Is AI Riding a One-Trick Pony?" <https://www.technologyreview.com/s/608911/is-ai-riding-a-one-trick-pony/>. (26 Feb. 2019).

Soni, Devin. 2018. “Supervised vs. Unsupervised Learning” <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>. (26 Feb. 2019).

Swalin, Alvira. 2018. “How to Handle Missing Data” <https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4>. (26 Feb. 2019).

van Craenenbroeck, Jeroen, and Jason Merchant. 2013. “Ellipsis phenomena”. In Marcel den Dikken, ed. The Cambridge Handbook of Generative Syntax. Cambridge: Cambridge University Press, 701–45.

Vicente, Luis. 2015. *Morphological case mismatches under sluicing*.

Xu, Zhixiang, Minmin Chen, Kilian Q. Weinberger, and Fei Sha. 2013. *An alternative text representation to TF-IDF and Bag-of-Words*.

Zinkevich, Martin. 2016. “Rules of Machine Learning. Best Practices for ML Engineering” <http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf>. (26 Feb. 2019).

Zwarts, Joost. 2005. *The case of prepositions: Government and compositionality in German PPs*.