Lorenz Leitner, BSc

Automatic Detection of Idiosyncratic Phrases as Features for Authorship Attribution

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, August 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

August 5th, 2020

Date

Signature

Abstract

People use different styles of writing according to their personalities. These distinctions can be used to find out who wrote an unknown text, given some texts of known authorship. Many different parts of the texts and writing style can be used as features for this. The focus in this thesis lies on topic-agnostic *phrases* that are used mostly unconsciously by authors. Two methods to extract these phrases from texts of authors are proposed, which work for different types of input data. The first method uses n-gram tf-idf calculations to weight phrases while the second method detects them using sequential pattern mining algorithms. The text data set used is gathered from a source of unstructured text with a plethora of topics, the online forum called Reddit. The first of the two proposed methods achieves average F_1-scores (correct author predictions) per section of the data set ranging from 0.961 to 0.92 within the same topic and from 0.817 to 0.731 when different topics were used for attribution testing. The second method scores in the range from 0.652 to 0.073, depending on configuration parameters. In current times, due to the massive amount of content creation on such platforms, using a data set like this and using features that work for authorship attribution with texts of such nature is worth exploring. Since these phrases have been shown to work for specific configurations, they can now be used as a viable option or in addition to other commonly used features.

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

People have different personalities and behaviors. These unique features are also called *idiosyncrasies*. They are reflected in many aspects of an individual's life, such as habits in normal life, including speech (gesturing, enunciations, choice of words, sentence structure, et cetera) and also writing, similarly to speech, but without the physical attributes. Particularly in writing, idiosyncrasies can be easy if laborious to spot by humans, if they are paid attention to. Machines can automate many human information retrieval, training and learning tasks, and may achieve better outcomes, and speed up the process. One of these unique idiosyncratic features in a person's writing is the stylistic choice of words and use of certain phrases, that are used less frequently by other people.

1.1. Hypothesis

The main hypothesis of this thesis is that different individuals use different words and phrases in their writing, according to their idiosyncratic traits. Based on that, it would be possible to identify authors of texts using the words and phrases that occur in the texts. This also evaluates how distinctive the phrases actually are, and if they can be used as features for this process, which is also called *authorship attribution* or *authorship identification*, that is attributing unknown texts to authors for whom texts are known, or similar processes such as *authorship verification*, which is verifying whether a text has indeed been written by a supposed author. This type of classification has been done in many ways before, but with many other features, such as low-level approaches via characters or word length, or higher-level approaches looking at sentence structures or whole author profiles and meta-data. Chapter 2 lists these other approaches. Intuitively it makes sense that a person differs from another under the condition of free speech or writing. Identifying and predicting authors based on these differences evaluates how valid

the hypothesis is and how suitable the approaches used for gathering the features are. For comparisons, a baseline can be defined with a naïve random classifier or another method of the current state-of-the-art. Using studies with human test subjects to form a baseline as done by Marujo et al. [2013] and Rexha et al. [2018] is deemed out of scope for this work.

1.2. Examples

The following lists some examples of such stylistic or idiosyncratic words and phrases as mentioned earlier in English, which is the language on which the focus lies:

- Words or phrases used mostly *unconsciously* such as “long story short”, “let me tell you”, “nonchalantly”, “up my alley”, “I hear you”, “props to you”, “I’m calling BS”, “bummer”, “for sure”, “I suppose” or “that being said”.
- *Regional idioms or slang* like “hella”, “wicked”, “bless your heart” ,“I reckon” or “neither here nor there”.
- More *archaic* words such as “behoof”, “froward” or “gallant”.
- *Internet-specific spellings* or abbreviation like “u” instead of you, “ymmv” (your mileage may vary) or “afaik” (as far as I know).
- *New terms* coming from younger generations that seemingly sprout up at random like “bruh” and “yeet”.
- *Errors*, in specific misuses or misspellings to which a person might be prone can indicate a distinct style of writing: “could/should/would of”, “could care less”, “supposably” and so forth.

1.3. Writing Genres in the Internet Age

The method of attributing authorship based on idiosyncratic words and phrases may work better or worse, depending on the source of *corpora* used, which are the bodies of text. If authors write on the same topic, ideally only the free-choice words should differ, that is the words that are not related to the topic. However, if the authors write formal texts such as articles in journals or newspapers, not many stylistic choices can be made, as they are dictated by the genre of text. And

if the topics differ, the topic words would make up the greatest difference. On the other hand, if the genre of writing contains highly unstructured text such as online-forums with no regulations, say the comment section of said newspaper articles, or forums dedicated to specific topics, the choice of stylistic words and phrases is much more free, and authors take part in it, even unconsciously [Argamon and Levitan, 2005]. In the age of massive amounts of data that is being produced on the Internet, the frequency of such texts is much higher than it used to be. This is why many earlier approaches regarding natural language processing and authorship attribution focused on works of literature [Stamatatos, 2009], rather than online-content. Therefore it makes sense to thoroughly test new approaches on these sources of data as well and continuously come up with new methods due to the ever changing nature and increase of online-data.

1.4. Use Cases of Phrase Extraction and Authorship Attribution

Generally speaking, why would phrase extraction and/or authorship attribution be useful?

Phrase extraction alone is often used to extract information from texts, for example to summarize texts, similarly how articles often include key words and phrases at the top of the document [Ahonen et al., 1998; Marujo et al., 2013]. This is also called *key phrase extraction*, as it extracts the *key* phrases of a text. This information can then be used for indexing and fetching fitting results during a search of documents, or recommending documents based on similar ones. This is mostly about the topic contents of texts, while this work focuses more on stylistic phrases.

Authorship attribution is used for many different purposes, some in the nature of historical literary research, such as settling debates whether some famous work of literature has been written by a specific renowned author or rather by someone else. In the current age, it is also used in more applied ways, for instance controlling for plagiarism, or in criminal law and forensic science, to gather information about threats of terrorism or hate crimes sent via online-platforms, or to identify individuals who harass others anonymously over the internet, or even to identify authors of letters like bomb threats. This can of only be done if some text data

written by known suspects exists. For example, an employee may be harassed by someone anonymously, however, it is suspected that the person behind the messages is someone known by the victim, like a colleague or acquaintance. Known written data can then be gathered and compared to the evidence of text of yet unknown authorship. In case there is only one suspect or candidate, and the known text is compared to the unknown, the process is called *authorship verification*. If there are multiple candidates, and the most likely one is chosen, it is referred to as *authorship attribution*. See also Section 2.1.4 for a detailed description of the nomenclature and differences between approaches.

1.5. Choice of Data Set

The data set used contains the corpus, which consists of comments gathered of users on Reddit[1], as the texts written there pertain to the unstructured usage described before, and the data is already labeled by author and topic and can include additional meta-data that may be of interest. The topic in this case would be a *subreddit* (which have recently began being called "communities" officially, however henceforth the former term shall be used, as it is more distinctive to the domain). Also from now on the terms "subreddit" and "topic" may be used interchangeably in this context, as it applies on Reddit in this way: Subreddits are sub-pages that limit discussions only to specific topics. The plethora of user-created subreddits and posts containing comments make Reddit an excellent choice for retrieving natural text. Either an existing data set can be used if a suitable one can be found, or new data can be scraped from Reddit via its Application Programming Interface (API)[2] and one of the Python clients with which to use it[3]. The issue with most existing data sets is that they are either so large that they are unwieldy to handle on a local machine, with comments being in the billions[4], and monthly or daily snippets taken from these huge sources are too horizontal in that they contain many comments by different users, but not many comments of a few distinct users. A better approach for this use case would be to take existing users, and get their comment history as far back

[1] https://reddit.com
[2] https://reddit.com/dev/api
[3] https://praw.readthedocs.io
[4] https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment

as they go, and then filter for subreddit. The selected users can also be screened in advance on which subreddits they frequent, as to get many relevant comments. Another possibility is to take the n top commenters of a subreddit, who can be ascertained either by peeking into a large data set mentioned earlier via Google BigQuery[5] or via some public statistics collection method like AssistantBot[6], for subreddits that make the collected data available. This can be done locally as well by using the Pushshift API[7]. Using PSAW[8], AssistantBot's results can be replicated locally. Scraping the data and using PSAW is the chosen approach in this work.

1.5.1. Amount of Data

Large amounts of consideration can go into the selection of the size of the data set, and the distribution of the data, that is documents per author. Some existing approaches are listed below, pertaining to similar areas, such as online media and shorter texts, in contrast to, say, historical literature work.

For instance-based approaches of authorship attribution (see Section 2.1.3 for an explanation about the different types of authorship attribution, in essence, instance-based approaches have multiple documents per authors), text blocks of size 200, 500 and 1000 were used by Hirst and Feiguina [2007]. Longer text blocks increased the accuracy.

Overdorf and Greenstadt [2016] collected 200 authors of Twitter and 100 authors of Reddit, each with 10,000 words in each domain, but used a random subset of 50 authors for the actual experiments, as anything beyond that was deemed too computationally expensive. They split the texts into 500 word documents. They also mention previous in-domain work [Afroz et al., 2014], which has shown that 4,500 words and 500 words are enough for training and test documents, respectively.

Ruder et al. [2016] collected data from different domains, for Reddit, they collected 9,266 authors, with a median of 216 documents each, and a median document size of 31. This is a much larger data set than most others mentioned elsewhere, however,

[5] https://bigquery.cloud.google.com
[6] https://github.com/kungming2/AssistantBOT
[7] https://github.com/pushshift/api
[8] https://github.com/dmarx/psaw

the classification was handled using Convolutional Neural Networks, which excel at this type of large-scale task.

Another online data set was collected by Suman et al. [2020]. They collected Tweets of 108 users, which was reduced to viable 34 authors after some data cleaning was done. The distribution of Tweets per user amounted to 500, with randomly selected 450 training and 50 test Tweets. They also mention the RCV[9] and PAN[10] data sets, which can be used for authorship attribution experiments, as reasons for the size of their data set, as RCV and PAN contain even fewer documents per author, about 50 and two or three, respectively.

Clark and Hannon [2007] used two to four authors in their tests, however, those were longer texts, ranging from about 120,000 to 180,000 words.

1.5.2. Data Preparation

Some data cleaning may be done, to prevent off-topic comments, which may occur at lower levels of a *comment-thread* or tree, which are consecutive replies of comments. They could be truncated at a specific level. However, the parameter `nest_level` is not available for all of Reddit's comments, and computing this value dynamically poses a too high computational expense. In practice, this supposed off-topic nature of deeper level comments does also not show up that often, and if, it does not appear to skew the topic distribution of an author, so this data preparation step can be omitted.

However, more important before doing any work on the data, is choosing a balanced set of authors and texts. If the topics and authors are not balanced, results can be tainted due to this, see Section 1.5.4.

For the very large dataset mentioned before, which is hosted on pushsift.io[11], there exists an API[12] which can be used to access and filter the data. For example to get all of a user's comments and filter them by subreddit:

[9] https://archive.ics.uci.edu/ml/datasets/Reuter_50_50
[10] https://pan.webis.de/clef19/pan19-web/
[11] https://pushshift.io
[12] https://github.com/pushshift/api

`https://api.pushshift.io/reddit/search/comment/?author=<user>&subreddit=<subreddit>`

By default, this only retrieves the 25 most recent comments, however the `size=<n>` parameter can be appended, or before and after dates can be used as well.

Since PSAW exists, a Python wrapper that provides utility functions for easier usage of this API, collecting, filtering and balancing data according to the requirements is a task that can be viably automated.

1.5.3. Subreddit Statistics for Candidate Subreddits

Some subreddits have more potential to be a source of comments, due to the fact that they pertain to either one topic or are general discussion-based. Also, they have a high number of comments, and commenters with many comments in one subreddit, as can be seen by AssistantBot's data collection in the respective subreddit's wiki page[13].

The following subreddits have public top commenters statistics produced by AssistantBot and have the previously described properties:

- *boxoffice*
- *brandonsanderson*, *Cosmere*, *Mistborn*, *Stormlight_Archive* (Clique of similar topics with the same users)
- *ScenesFromAHat*
- *livepd*
- *JusticeServed*
- *HomeworkHelp*
- *classicwow*
- *TrueCrimeDiscussion*
- *Warthunder*

These subreddits and their wiki pages can be explored as examples of what the statistics of a subreddit may look like, for a better understanding of the nature of different subreddits in advance, before diving more into detail in later sections.

[13] Example: https://www.reddit.com/r/boxoffice/wiki/assistantbot_statistics

What can be used to generate similar statistic on-demand is prawtools[14]. For example: `subreddit_stats -v -c 50 tifu 30` generates statistics of the *tifu* subreddit over 30 days including the top 50 commenters, although they are sorted by gained *karma*, which is the sum of received upvotes and downvotes, not frequency, which would be the ideal sorting method.

After some more in-depth research and generating the statistics locally for different subreddits, the list of subreddits to be used was settled on:

- *AmItheAsshole*
- *askreddit*
- *books*
- *boxoffice*
- *classicwow*
- *games*
- *gaming*
- *HomeworkHelp*
- *MakeNewFriendsHere*
- *movies*
- *news*
- *nextfuckinglevel*
- *politics*
- *teenagers*
- *tifu*
- *todayilearned*
- *unpopularopinion*
- *worldnews*

As it turns out, some of these subreddits are more suited to the task and others less so, depending on the type of text and language that is used in them. See Sections 1.5.5 and 4.4 for more information, which subreddits and why, and also descriptions of all of these subreddits in the list. These multiple subreddits have been chosen in order to test their suitability to the task.

[14] https://github.com/praw-dev/prawtools#subreddit_stats-examples

1.5.4. Class Imbalance Problem

Most existing methods use a relatively even distribution of the training corpus, that is the authors have similarly numbered and sized corresponding texts. Some existing approaches fare worse with imbalanced cases, such as as the initial *Common N-Grams* (CNG) approach [Kešelj et al., 2003]. Given that most existing approaches assume a balanced input set though it is fair and beneficial for the input set of this work's approach also be balanced. Some data cleaning may have do be done to ensure this, especially if data is retrieved from a highly unstructured source such as online-forums. Data cleaning methods may include splitting longer texts into shorter chunks or combining shorter texts into longer ones.

1.5.5. Topics

Ideally, any stylistic choices unique to authors should be the main focus of topic-independent authorship attribution. Therefore the evaluation corpus should be controlled for topic, say all authors write about the same topic. Otherwise an author could be unique in that a specific topic is favored, in contrast to other authors who frequent other topics. In specific, more topic-unique words would be differentiated instead of stylistic choices. Limiting the evaluation corpus to just one topic prevents this, or at least limiting comparisons or different runs to the same topic each suffices, if multiple topics are present. This can be easily done if the data is retrieved from online forums which offer topic-categories, such as Reddit and its *subreddits*. Again, subreddits are sub-pages that limit discussions only to specific topics. The choices of topics themselves are of interest, since they may lend themselves better or worse to the task. For example, a primarily joke-based subreddit such as */r/dankmemes*[15,16], where mostly jokes and short well-known word-combinations are reiterated, will most likely fare far worse than a subreddit like */r/tifu*[17], in which stories about mishaps of the users are told in more length, with comments often telling similar stories. The comparison and evaluation of the accuracy with different topics or subreddits might therefore also be of interest.

[15] */r/* declares the name of a subreddit.
[16] https://reddit.com/r/dankmemes
[17] https://reddit.com/r/tifu

2. Related work

Authorship attribution is an old topic, going back as far as the 19th century, which has been approached from a multitude of angles and used for a various applications. A comprehensive overview of modern approaches is given by Stamatatos [2009]. Phrase extraction too is a common topic, although most often used in different settings, such as key-word extraction for information retrieval [Riloff and Lehnert, 1994; Ahonen et al., 1998; Zhang et al., 2005; Marujo et al., 2013].

2.1. Background

The main underlying technologies and methods come from the fields of Artificial Intelligence and Natural Language Processing (NLP), as well as Information Retrieval and Pattern Mining or Text Mining [Witten, 2004].

2.1.1. Tools

There are many tools that can be used for NLP in different programming languages, which implement some of the well-known methods and algorithms, and in general tools that implement certain solutions to recurring problems. Also the initial data retrieval is made easier by using certain APIs that grant utility functions to access data on web sites in more convenient manners.

Such tools for Python[1] that are relevant and useful in the cases of phrase extraction and authorship attribution:

- *NLTK* - General NLP tools for Python[2] [Loper and Bird, 2002]

[1] https://python.org
[2] https://nltk.org

- *scikit-learn* - Machine Learning library for Python[3] [Pedregosa et al., 2011]
- *SPMF* - Data Mining library for Java[4] [Fournier-Viger et al., 2016]

These are general tools that can be used in these fields, and also were used in this thesis. The more specific tools, and all used versions and their numbers, are listed in Section 3.2.

2.1.2. Phrase Extraction

A key process to identifying authors can be extracting often-used phrases from an author's text. These could then be compared to phrases in a text of unknown authorship versus phrases from another author. For example, if an author uses a particular phrase more than another, and that phrase also occurs in the unknown text, it is an indication that this author may have written that text.

How phrases are extracted depends on the used approach. It could either be done using linguistic features, or algorithms based on pattern mining, or manual rules.

2.1.2.1. Linguistic Features

One approach for the retrieval of phrases would be to gather all the *word n-grams*, which are sequentially used words, from an author's text, and save the usage count of each. Then, sort this list according to frequent usages, and the most-often used phrases will be apparent. This can be enhanced by adjusting the n values of the used n-grams, or including or excluding certain grammatical phrase structures, via *Part-of-speech* (POS) tagging, which is the process of assigning tags to words which correspond to their grammatical usage, such as nouns, verbs, adjectives, et cetera.

2.1.2.2. Pattern Mining

Pattern mining is the process of extracting patterns from data and may be used as a more general approach without linguistic features. In specific, *sequential pattern*

[3] https://scikit-learn.org
[4] http://www.philippe-fournier-viger.com/spmf

mining focuses on sequential patterns in sequential data, meaning the items in the data are ordered in some fixed way. This could be time-series data, or DNA, or a sequence of actions, or, as in this case, ordered words in text. A sequential pattern is then a subset of the overall sequence(s) that is distinct in that it appears more often than other subsets. A comprehensive overview of different sequential pattern mining algorithms was done by Fournier-Viger et al. [2017].

2.1.2.3. Phrase Weighting

During or after the extraction of words and phrases, they can be weighted according to their importance. These weights can be saved in addition to the phrases and their other attributes, mainly their frequency of occurrence. Why does a weighting scheme matter for this use case? Some phrases may occur often because the topic that is being written about contains them, but others stem from pure stylistic and idiosyncratic choices from the author. Therefore, if the authors write about the same topic, the phrases that occur for all of them frequently can be disregarded, however, the ones that are often used by one author, but not by others, are a good indication for classification, and should be weighted higher. Ideally these phrases should also be valid if the author writes about different topics, but the extraction and attribution methods are not as straightforward in this case, see Section 2.1.2.4. A weighting scheme such as the the *term frequency-inverse document frequency* (tf–idf) statistic may be used, which signifies importance of a word in a document, but is offset by the occurrence of that word in the overall corpus of documents [Ramos, 2003].

2.1.2.4. Situation of Author Number and Topics

There are multiple situations in which the phrase extraction can occur, and the used phrase extraction (and classification) approach can differ based on that.

One author and one topic Only one author is given and that author only writes about one topic. Sequential pattern mining algorithms can detect patterns of phrases over the whole concatenated text, but these will also include topic-related words, not only topic-independent stylistic words and phrases. Splitting up the concatenated text into smaller chunks and using n-gram tf-idf would yield no

information, since each smaller chunk comprises essentially the same information as the larger concatenated text, given the same author and the same topic.

One author and multiple topics One author is given but that author writes about multiple topics. If the topics are evenly distributed over a concatenated text (if not ordered), sequential pattern mining algorithms should recognize only patterns that stem from idiosyncratic stylistic choices, without including topic-specific words, since the author's writing style should be present regardless of the topic that is being written about. Splitting the concatenated text into chunks segmented by topic, and using n-gram tf-idf weighting, this should actually yield the topic key words and phrases with high weights, since these are the ones that differ per topic, while the stylistic phrases stay consistent. Using the inverse of these weights may provide the needed weights for the idiosyncratic phrases, but this approach is untried.

Multiple authors and one topic Multiple authors are given and these authors only write about one topic. Using n-gram tf-idf (that is n-gram tf *and* n-gram idf) in this case, with a *document* being the concatenated texts of an author, it should identify and highly weight the correct words and phrases, that is the author-specific stylistic ones, since these are the ones that differ per document, as all authors write about the same topic.

Multiple authors and multiple topics Multiple authors are given but they write about multiple topics. Per individual author, sequential pattern mining could again be used, just as described in the situation of one author and multiple topics. For any n-gram tf-idf weighting, the topics cannot be handled indiscriminately, for they may skew the weights. The distribution of topics must also be taken into account. If an even distribution of authors and topic can be achieved, the method described in the situation of multiple authors and one topic can be used, but on a per-topic basis. For example, for Authors $A_1 \dots A_n$ and Topics $T_1 \dots T_m$ where each A_i writes about all $t \in T_1 \dots T_m$, the n-gram tf-idf weighting described before can be used for documents about one t each. Since this way, each tf-idf weighting/comparison only pertains to one topic, but multiple authors, the weights should still be correct and favor stylistic words and phrases. Authors who do not write about the same topic cannot be included in that comparison, but their document would be "empty" then

```
Intersection using subreddit "/r/movies":
>>> Common top commenters with /r/todayilearned: 1
>>> Common top commenters with /r/worldnews: 2
>>> Common top commenters with /r/books: 1
>>> Common top commenters with /r/boxoffice: 10
>>> Common top commenters with /r/games: 2

Intersection using subreddit "/r/worldnews":
>>> Common top commenters with /r/askreddit: 1
>>> Common top commenters with /r/todayilearned: 1
>>> Common top commenters with /r/movies: 2
>>> Common top commenters with /r/news: 7
>>> Common top commenters with /r/gaming: 1
>>> Common top commenters with /r/books: 1
```

Listing 2.1: Sample output of non-empty intersections of the top 100 frequent commenters of different subreddits.

in any case. In other words, it is the method described in the previous paragraph, but multiple times over a range of topics.

However, it is difficult to gather a data set that fits a criterion like this. Multiple authors who write a lot about the same different topics are rare. Essentially what is needed for this: Authors $A_1 \dots A_n$ and topics $T_1 \dots T_m$, and all authors are evenly distributed top commenters in these topics. This is unlikely to be found in sources such as Reddit, as when and if an author is a top commenter in a subreddit, the author tends to stay within it. If multiple subreddits are frequented, the distribution of comments is such that no statistical significance of one subreddit is found, but much fewer comments spread across multiple subreddits. Intersecting the sets of the top 100 frequent commenters of multiple subreddits yielded very few matches. Single top commenters rarely are in the top 100 commenters of two subreddits of disparate topics and multiple but also not many top commenters sometimes are in the sets of subreddits pertaining to very similar topics, such as */r/news* and */r/worldnews* or */r/movies* and */r/boxoffice*. Listing 2.1 shows and example of these observed intersections.

An easier way to test whether an author's writing style is consistent across topics is to collect author texts about one topic mainly, train and weight on this set, but

when testing, also include texts about different topics. The trained stylistic phrases should still occur and classification should still be possible. This is one of the testing methods used in this work.

Also worth noting, a subreddit itself contains somewhat different topics. Even if the subreddit is limited to for example /r/movies, the posts within will present slightly different topics. This may provide the needed middle ground between a hard limitation of topic and difference of topic. See also Section 1.5.5 about the nature of topics.

2.1.3. Authorship Attribution Methods

There are commonalities and differences between different methods of the attribution of authors to texts.

Common to all methods is the input:

- *Training corpus*: A set of candidate authors and a set of documents/texts with known authors (from the set of candidate authors).
- *Test corpus*: A set of documents/texts of unknown authorship, which shall be attributed to authors of the set of candidate authors. These "unknown" authors are of course still labeled in a research setting, as to use for verifying whether a classification is correct.

The unknown texts are then compared with the known texts via some similarity/distance measure, and the most likely author can be classified.

What can be different is how the individual known texts are treated. They may be taken as individual documents, so one author has multiple corresponding documents, or concatenated into one long text file, so one author has one document containing all corresponding input texts. The latter disregards differences of the individual texts, as the separations may not be distinguishable after the concatenation. These approaches are called *instance-based* and *profile-based* respectively [Stamatatos, 2009]. Of course, a hybrid approach may also be used, combining both in some way [Van Halteren, 2007].

Within these two main categories, additional different methods can be used. The following gives an overview of existing methods [Stamatatos, 2009].

Profile-based approaches:

- *Probabilistic models* [Zhao and Zobel, 2005; Sanderson and Guenter, 2006]
- *Compression models* [Marton et al., 2005]
- *Common n-grams models* [Kešelj et al., 2003]

Instance-based approaches:

- *Vector space models* [Sebastiani, 2002]
- *Similarity-based models* [Burrows, 2002]

Another difference about profile-based and instance-based approaches, as can be seen by this list, is if and how they use a training phase, or just calculate distances without machine learning, using some similarity function such as Euclidian distance, cosine similarity or Pearson correlation. Instance-based approaches like vector space models tend to train a classification model, with some statistical or machine learning algorithms, while other approaches like the similarity-based models or some profile-based models do not. Also, for example, the k-nearest neighbor classification does also not really "train" either. In other words, the *training phase* of some profile-based approaches is only comprised of gathering the features from the texts, which in this case could be phrases. This profile-based approach is also how this work constitutes its authors.

2.1.4. Authorship Identification, Verification, and Profiling

There are processes that sound similar in name, but are different in nature. The following describes their differences.

Authorship identification is the process of identifying the author of a text of unknown authorship. It is synonymous with "authorship detection" and "authorship attribution", the latter of which is used mainly in this work, however the former terms are also used sometimes in the literature. How this process works in detail is described in Section 2.1.3.

Authorship verification is the process of verifying if a given text is written by an author. There is only one author with known texts and one unknown text. The question is whether the unknown text has been, too, written by the known author.

Authorship profiling is the process of creating an author profile describing characteristics of the author based on known authored texts. Characteristics can include age, gender, personality, et cetera.

2.1.5. General Issues and Information

A few general nuances regarding the mentioned processes, the general field and related work come to mind, such as the intentional omission of the removal of stop words, and the case of the elusive author, which are described below.

2.1.5.1. Function Words

Argamon and Levitan [2005] note that function words, that is stop words such as articles, pronouns, prepositions like “the”, “it”, “on”, et cetera, are suited for extracting stylistic information from text. They are often excluded in topic-based classification tasks, as they provide no semantic information, but should be included in style-based classification tasks. These function words are used mainly unconsciously by authors, therefore representing a good indication of their writing style, regardless of the topic. Thus it is most likely beneficial to include these in stylistic phrase-based classification.

2.1.5.2. Open-World Problem

The *open-world problem* describes the case when the author of an unknown text is not in the set of the known authors, which is the training set. If that is not taken into account, the classification might result in a wrong author, even if the likelihood of a match is low, it still will be the most likely match. An option would be to use a “no match found” output in such a case. A verification step that estimates the likelyhood of the most likely match being the right author can be added, which then would decide whether to use this special output [Stolerman et al., 2013].

2.2. State of the Art

Some existing approaches try to solve similar problems in different ways. Other methods can differ in how they treat texts and authors as a whole, and what methods for classification are used. These differences have already been described in Section 2.1.3. Other differences may be the features that are selected for classification. All the different features used in existing scientific work are listed in Section 2.2.3. The following section gives an overview of existing scientific work in the field that achieve a high performance and accuracy and a more detailed look into some of the publications.

2.2.1. Methodology

Researching relavant scientific literature was conducted using Google Scholar[5] which is an index of mostly free-access scientific papers, but also links to other collections, such as the Association of of Computing Machinery (ACM)[6], as well as IEEE Xplore[7], a library referencing papers of the IEEE Computer Society, and SpringerLink[8]. Access to papers behind a paywall is granted for connections stemming from universities such as Graz University of Technology, therefore a virtual private network (VPN) connection was used to gain access.

Relevant search terms used are:

- *Phrase Extraction* - The process of extracting phrases from text
  - *Key Phrase Extraction* - Phrases summarizing topic of a text
  - *Stylistic Phrase Extraction* - Phrases used for stylistic reasons
  - *Idiomatic Phrase Extraction* - Language-specific idioms
  - *Idiosyncratic Phrase Extraction* - Author-specific phrases
- *Authorship Attribution* - Attributing authors to texts of unkown authorship
- *Stylometry* - Linguistic style of written text
- *Idiosyncrasy* - Unusual features unique to a person
- *Data/Pattern Mining* - Discovering (hidden) patterns in data sets

[5] https://scholar.google.com
[6] https://dl.acm.org
[7] https://ieeexplore.ieee.org
[8] https://link.springer.com

Overall	Relevant	Read in detail
108	66	6

Table 2.1.: The numbers of literature found: Overall, deemed relevant, and studied in detail.

- *Frequent/Sequential Pattern Mining* - Mining frequent patterns in sequential (ordered) data series

Another rather obvious way of finding relevant literature is looking at the references of already found papers and papers that cite existing papers. Many mention and summarize existing relevant papers in their own Related Work chapters, therefore it is simple to pick out further interesting resources.

2.2.2. Existing Scientific Work

Table 2.1 shows a summary of all the literature that was found in relation to this work with related search terms, how many have been deemed relevant, and how many have been studied in more detail, some of which are described further here.

2.2.2.1. Large Scale Authorship Attribution

Using corpora consisting of orders of magnitude more authors and texts than what has been dealt with in older works, has been approached using Convolutional Neural Networks (CNN) [Ruder et al., 2016]. This paper deals with large amounts of data using multiple datasets such as online forums like email, IMDB, Blogs, Twitter and Reddit, with numbers of documents ranging from 80,000 to 2,000,000, respectively. An interesting detail is mentioned related to Reddit, where the accuracy can be inconsistent due to the nature of specific posts or comments, that is "documents" by authors. The post length is often very polarizing, it might be very short due to comment-chains consisting of one-word replies, to iterate some common community-specific joke, whereas other posts may be rather long-winded detailed explanations or rambling rants. The very short posts do not lend themselves well for the intended analysis. In contrast to that is Twitter, where the standard deviation of post lengths is much lower, due to the fixed upper limit of characters a post can contain (280). CNNs are suited to the task at hand, since they excel at

extracting information from patterns such as images in computer vision, or text, as it is used here. Different variants of the CNN model have been used, with different input channels, such as generic character-level n-grams, words, or a combination of them.

2.2.2.2. Cross-Domain Authorship Attribution

This topic pertains to texts from authors across multiple domains. One author may have written in different domains which dictate genre, such as online forums, blogs or emails. Usually, machine learning approaches assume that the training set and test set stem from the same sources. In certain real-life cases, this may not be the case. For example, an unknown author could badmouth his company on Twitter, however the company does not have a training set of Twitter accounts of their employees, but they do of their emails. So the training set in this case would be emails, while the test set, that is the texts of potential unknown authors that need to be uncovered, is Twitter texts of suspects, which differ inherently in style from emails, regardless of authorship. Most state-of-the-art approaches assume same domains though, and fare worse in a cross-domain application, as shown by Overdorf and Greenstadt [2016], who also propose improved approaches to deal with this type of situation.

2.2.2.3. Multi-Modal Content

This type of content is not only comprised of text, but also of media such as (animated) images, videos, audio, emojis, et cetera, which all can be used as features characterizing authorship. Their existence and validity depends on the source of data of course, highly present for example in posts on Twitter. *Emojis*, which are image-representations of emotions, carry valid information that can be used for tasks like extracting sentiment, and since people have their own preferences of which emojis to use, and how to use them, they can be a valuable asset for tasks like authorship verification on platforms such as Twitter, however, not in all cases or with all authors [Suman et al., 2020].

2.2.2.4. Classification With Synonym-Based Features

An approach proposed by Clark and Hannon [2007] for weighting words more or less based on how many *synonyms* they have. Synonyms are different words that mean the same thing. Based on that, an author has freedom to choose different synonyms for one meaning, therefore indicating a stylistic choice. This means if a word has many synonyms, it should get a higher weight for classification purposes than a word with fewer or no synonyms. With a few more nuances, like accounting for the idiomatic use of a word in a common language, and *stemming*, the process of reducing word variations to their base form, in a variation of this model, considerable results were achieved. As it turned out, stemming actually has a negative impact, albeit a small one, on the results in this approach. This follows in line with the fact that the cut-off suffix itself can be used as a feature for authorship attribution [Muhr et al., 2010].

2.2.2.5. Sequential Pattern Mining

Not inherently related NLP or to authorship attribution but with a much more general potential application, this topic focuses on discovering patterns in data that is ordered in some way. See also Section 2.1.2.2 for a general description of pattern mining. Fournier-Viger et al. [2017] give a comprehensive overview of different sequential pattern mining algorithms and their use cases. All conventional sequential pattern mining algorithms produce the same result, given the same input and parameters, such as the desired minimum frequency or *support* of the frequent patterns. This is because sequential pattern mining is a problem with a fixed solution. The algorithms differ mainly in how they achieve this result, and due to that also in their performance. For example, differences can be if they explore patterns depth-first or breadth-first, their internal data structures, how supersequences are created coming from shorter sequences in the exploration process, and how the minimum support validity among their found patterns is counted. Seminal examples of such algorithms are GSP [Srikant and Agrawal, 1996], SPADE [Zaki, 2001], SPAM [Ayres et al., 2002], and PrefixSpan [Pei et al., 2004]. CM-Spam and CM-Spade [Fournier-Viger, Gomariz, Campos, et al., 2014] are some of the fastest of these standard algorithms according to Fournier-Viger et al. [2017].

Types of Sequential Pattern Mining Algorithms Also of note are algorithms that produce a more concise limited subset instead of all frequent patterns. This can be beneficial because it increases performance, improves human readability of the results, and can in some cases even improve classification accuracy if the patterns are used for that task.

For example, *closed* sequential patterns are the largest frequent patterns in the set. They provide a lossless representation of all sequential patterns, as those can be generated again from the closed patterns. They may lend themselves well to extracting the most frequent longest phrases of authors, since they exclude sub-phrases of the same support value. This can and has been done manually before by simply reducing a list of high rated patterns to only their longest supersequences after the results from a conventional algorihm are gathered, but if it is built into the algorithm at run time, it can reduce the needed overall time. This holds true only for a phrase dictionary though, as for this the longest phrases are most interesting. Shorter sub-phrases of equal importance can still be valuable for classification purposes, even though in some cases these concise subsets improved classification.

Similar to closed sequential patterns are *maximum* sequential patterns, which represent an even smaller set than closed patterns, however, they are not lossless, as the support cannot be regenerated again for all patterns. Finding the frequent longest common subsequences in sentences is an application of maximum sequential pattern mining algorithms.

CloSpan [Yan et al., 2003] and BIDE [Wang and Han, 2004] are examples of closed sequential pattern mining algorithms, as are ClaSP [Gomariz et al., 2013], CM-ClaSP [Fournier-Viger, Gomariz, Campos, et al., 2014] and CloFAST [Fumarola et al., 2015], which aim to achieve better performance. VMSP [Fournier-Viger, Wu, et al., 2014] is an example of a maximum sequential pattern mining algorithm.

Sequential Pattern Mining Algorithm Constraints In addition to algorithms built for reducing the output set, different constraints can be included as well, such as *gap constraints*, which limit the intermediate items of a subsequence that are not part of the supersequence, *item constraints*, that is which items themselves should or should not appear in the results, and *length constraints*, controlling the minimum and maximum length of the frequent patterns. Regarding gap constraints, somewhat counterintuitively, in a general definition of a subsequence of a supersequence,

gaps are allowed. For example, for the sequence $\langle\{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}\rangle$, the subsequence $\langle\{b\}, \{f, g\}\rangle$ is contained in the original sequence, while the sequence $\langle\{b\}, \{g\}, \{f\}\rangle$ is not. (Example taken from Fournier-Viger et al. [2017].) As can also be seen, sequences even work with item sets, but these can also be disregarded for an application with ordered words as items only.

Using these constraints during the process of an algorithm can greatly reduce its run time and space requirements. Additionally, for specific purposes like mining consecutive word phrases from sentences in text data, gap constraints are essential. Setting the gap constraint to a maximum of 0 or 1, depending on the implementation, ensures that only complete consecutive words are mined and no words are skipped. If a human-readable phrase dictionary like the one in this thesis is to be built, this is essential. Otherwise, without a gap constraint, words are skipped, and more common patterns are found, for instance the pattern *the the*, which occurs often because *the* is often followed by *the* again later in a sentence, however, of course with words between the two occurrences. These intermediate words comprise the gap.

Another way of reducing the output set is instead of letting the user choose a minimum support, let a *top-k* be chosen, meaning only the top k most frequent patterns will be reported. This is easier for users who do not know which minimum support value to set, due to lacking knowledge of the database, which can result in too many or too few patterns in the output for the user's liking. Only by trial and error can a user come up with proper values for the minimum support by hand, therefore *top-k* variants simplify this process. Examples for top-k sequential pattern mining algorithms are TKS [Fournier-Viger et al., 2013] and SKOPUS [Petitjean et al., 2016].

2.2.3. Features

Other methods of authorship attribution include features such as lexical/character, syntactic, semantic, topic-specific and application-specific features. Examples in scientific work for these types of features are shown in Table 2.2.

A comprehensive list is provided by Abbasi and Chen [2008], who offer an often used *writeprints* feature set, and also Stamatatos [2009].

Type of feature	Specific examples
Lexical/Character	Word counts/frequencies [Sebastiani, 2002]
	Word n-grams [Coyotl-Morales et al., 2006; Sanderson and Guenter, 2006]
	Character counts [J. Li et al., 2006; Grieve, 2007]
	Character n-grams [Peng et al., 2003; Kešelj et al., 2003]
	Writing errors [Koppel and Schler, 2003; Kern, 2013]
	Unique vocabulary [De Vel et al., 2001]
Syntactic	Part-of-speech (POS) [Baayen et al., 1996; Zhao and Zobel, 2007]
	Chunks [Stamatatos et al., 2000; Stamatatos et al., 2001]
	Sentence/phrase structure [Karlgren and Eriksson, 2007]
Semantic	[Argamon et al., 2007]
Topic-based	[Zheng et al., 2006]
Application-specifc	[Zheng et al., 2006]

Table 2.2.: Examples of what types and specific features other methods of authorship attribution use.

3. Method

This chapter describes the underlying approaches used to achieve the desired goals of phrase extraction and authorship attribution. Using these descriptions, it is intended that the system can be replicated independently. The following is separated into a Concepts chapter, which describes the methods used in a more abstract way, while the Implementation chapter takes a more granular look at the technologies that were used.

Figure 3.1 shows the overall pipeline of connected stages for the processes of phrase extraction and authorship attribution. Each output of one step is the input for the next step. It can be used as a point of reference as to where in the process each description of the stages of the overall process fits. The steps themselves are described in more detail below.

3.1. Concepts

The overall concepts surrounding phrase extraction and authorship attribution are described in Chapter 2, in specific in Section 2.1.2 and Section 2.1.3, respectively. Here, the methods that were actually used are described.

3.1.1. Phrase Extraction

As mentioned in Section 2.1.2, multiple methods to extract phrases from text can be used. This is related to the situation which the data set poses, as discussed in Section 2.1.2.4. As the gathered data comprises one topic, with multiple authors, but this multiple times, two main approaches can be used. In other words, the gathered data consists of one subreddit and its top 100 most prolific authors (and this for

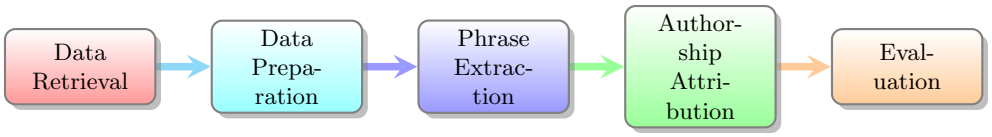

Figure 3.1.: Flow chart displaying the overall pipeline for the process of phrase extraction and authorship attribution. This is an abstracted view. The implementation may use some interspersed exploratory data analysis steps to get a sense of the data in its current state and data cleaning steps if necessary to clean the data to a desired state.

multiple subreddits). From these authors, the top five subreddits in which they contribute have also been gathered. This means a linguistic n-gram tf-idf approach can be used for the situation with one topic, multiple authors, that is the top 100 authors of a subreddit. A sequential pattern mining approach can be used for the situation of one author, multiple topics, that is the top five subreddits of a user.

3.1.1.1. Method 1: N-Gram TF-IDF

In this method, one subreddit from the retrieved data is chosen, and its 100 most profilic authors/users and their documents/comments are retrieved. ("Author" is synonymous to "user", and "document" is synonymous to "comment" in this context.) Each user's comments within that subreddit are concatenated to one long document, then all user documents are balanced to a uniform length, to prevent imbalances. The start and end positions of a user's used text are saved in a meta file, so the unused texts can be used for further stages.

Using the now balanced user documents the n-gram tf-idf scores of all n-grams for users are calculated. *tf-idf* stands for "term frequency-inverse document frequency", so the tf-idf weight of a term is its "importance" to a specific document, in this case a user's whole document, that is the importance to the user itself. It is the frequency of a term in a document but offset by the occurrences in all documents. In this case however, *term* does not refer to a single word, but multiple consecutive words, or n-grams, that is phrases. The calculation works exactly the same as with single words, but multiple permutations of n-grams need to be used as terms. For example, in the sentence "This is a sentence.", if bi-grams, or two-word phrases are used, each tf-idf calculation is done for the phrases "This is", "is a", "a sentence". The

lower bound and upper bound for n-gram ranges can be set to a desired minimum and maximum length of phrases. A simple tf-idf weight can be calculated with:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

, with $w_{i,j}$ being the weight of term i for document (user) j, $tf_{i,j}$ the number of occurrences of term i in document j, df_i the number of documents containing term i and N the total number of documents.

Some more nuanced weighting schemes can be used and are provided by many programming libraries. For example, *sublinear tf scaling* was used to exclude some unwanted terms. For instance, in the /r/AmItheAsshole subreddit, many users shared a common top phrase "nta you", because most users start their comments with this phrase in this subreddit. (*NTA* means "Not the asshole", as this subreddit is used to judge whether people behaved in a mean way or were justified in their behavior.) Using sublinear tf scaling, this and similar shared top phrases can be excluded. In theory, sublinear tf scaling is commonly used to lower the weight of terms that occur many times in a document, as it is unlikely that a term that occurs 20 times carries 20 times the weight of the same term in a document where it appears only once. Essentially, the weighting for the term frequency is replaced with $1 + \log tf$. In this case, it works perfectly for this application as well.

In any case, as discussed in Section 2.1.2.4, as all the users share this one topic and only this topic is used for the calculation of the tf-idf weights, the most important phrases of a user are those which the user uses more often than other users, filtering out many topic phrases, but retrieving topic-agnostic phrases that pertain to a user's specific style of writing. All the phrases are included, there is no threshold to exclude phrases below a certain weight. The very low-weighted phrases however will not have a large effect on any calculations.

Figure 3.2 shows a visual overview of the process of phrase extraction using this first method.

3.1.1.2. Method 2: Sequential Pattern Mining

This method differs from Method 1 in the situation of topics and users (Section 2.1.2.4). Now, instead of multiple users and one topic, like in Method 1, one user

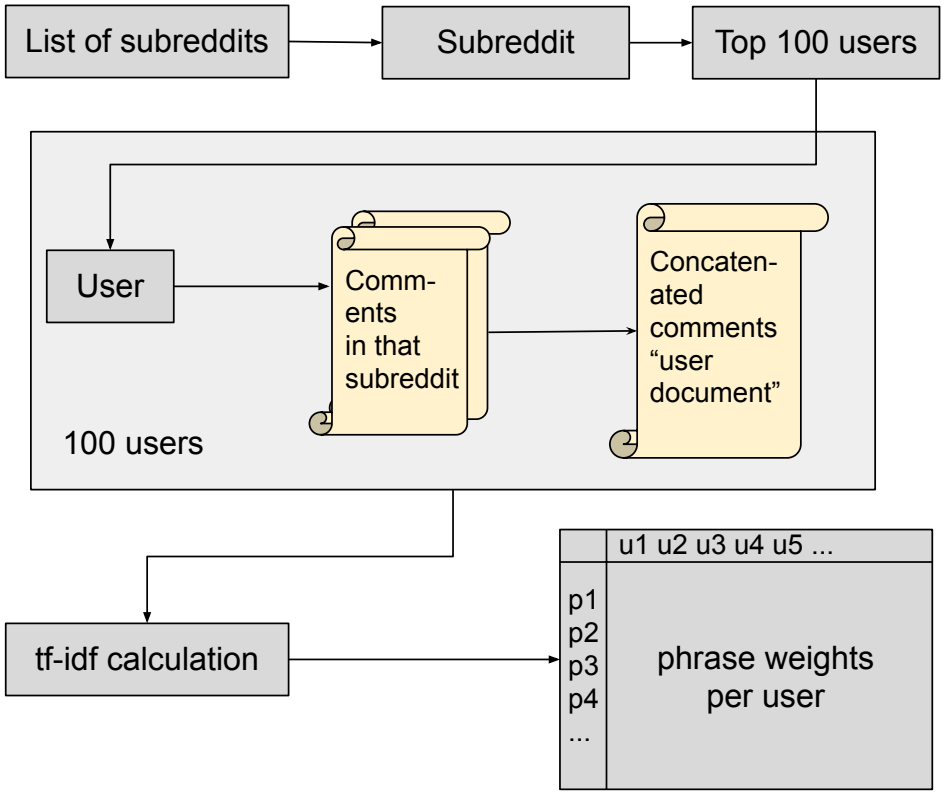

Figure 3.2.: Diagram displaying the process of phrase extraction using Method 1. First a subreddit is chosen from the list of selected subreddits. Out of that, its top 100 most prolific users are selected. For each user, the user's comments in that subreddit are concatenated to one long document. Balancing of document lengths is also done in between these steps. With all user documents concerning this one topic, the tf-idf calculation is done, resulting in a sparse matrix of all overall phrases with weights for each user.

and multiple topics is chosen. As five subreddits in which a user most contributes have been retrieved for each user, these serve as the topics. For each user, the user's comments in these five subreddits are concatenated, but again balanced in length. The shortest subreddit is chosen as a minimum length and a random range of that length is retrieved for each other subreddit. The start and end positions are saved in meta files so the unused text can be used for other purposes. On this balanced and concatenated user text a sequential pattern mining algorithm is used. A hard requirement for this algorithm is that it supports gap constraints, as otherwise non-consecutive words will be retrieved. The gap constraint needs to be set to 0 or 1, depending on the implementation, so that no words in sentences are skipped. Also beneficial is the ability of the chosen algorithm supporting lower and upper bounds for the pattern lengths. Not only can limiting the pattern lengths increase the speed of the process and lower the memory requirements dramatically, depending on the algorithm, it also produces a more concise output set of found patterns. As extremely large "phrases" with maybe only a support of one sentence in a user's overall text are unlikely to be desired to be found. See Section 2.2.2.5 for a detailed explanation about sequential pattern mining and the succeeding paragraphs about (gap) constraints.

As sequential patterns are repeated sequences in a longer sequence, in this case repeated phrases in a longer text, the only repeated sequences that should be found by a sequential patter mining algorithm in a user's text consisting of multiple topics of the same length should again be the user's topic-agnostic idiosyncratic and stylistic phrases.

Figure 3.3 shows a visual overview of the process of phrase extraction using this second method.

The minimum subreddit lengths mentioned before, below which subreddits of users are disregarded, is set to 50,000 characters for both methods.

3.1.1.3. Post-Processed Output

For both methods, in addition to the full and raw output of the tf-idf calculation and the sequential pattern mining algorithm output, a post-processed and cleaned output is produced. The raw tf-idf output is one very large sparse matrix containing all users and possible phrases. This is rather illegible for a human reader. Similarly,

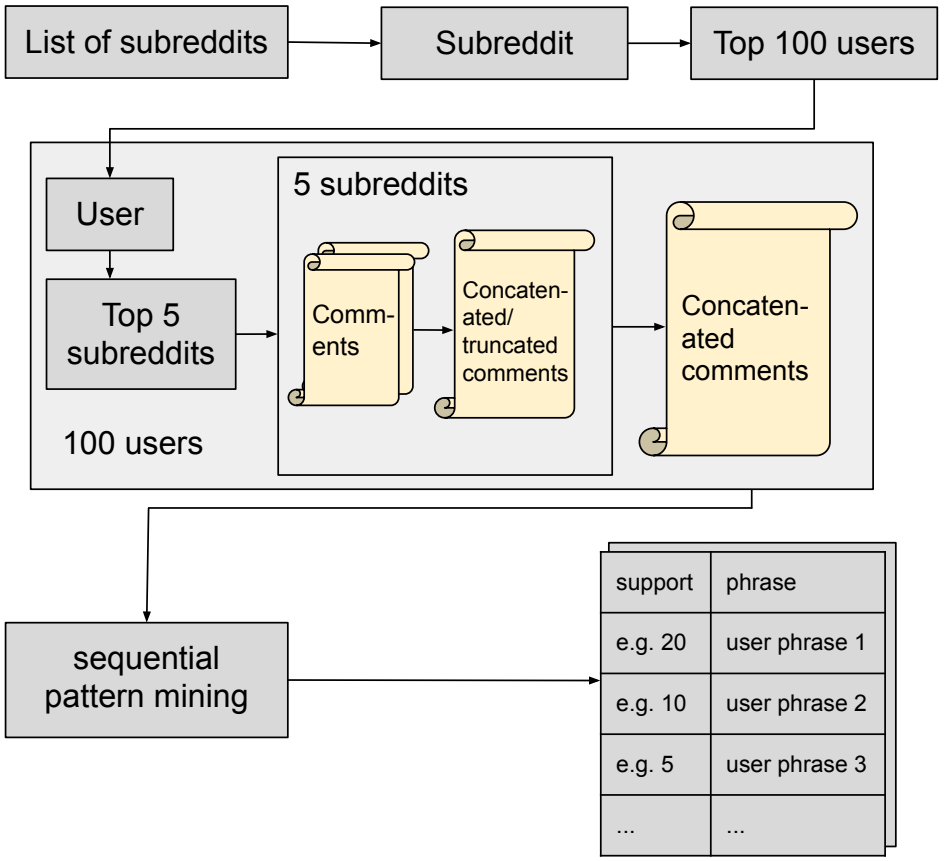

Figure 3.3.: Diagram displaying the process of phrase extraction using Method 2. First a subreddit is chosen from the list of previously selected subreddits. For each of its top 100 commenters, the comments of the top five subreddits of a user are retrieved (which includes the initial subreddit), balanced and concated into on long user document. On each user document of multiple topics, a sequential pattern mining algorithm is run which produces the most frequent phrases for that user. The "support" column in an output table displays in how many sentences of a user this phrase is contained.

the raw sequential pattern mining output may also contain far more phrases than a human can process. Due to this, a more concise "(top) phrase dictionary" is produced for each user, that includes only phrases particular to a user (in contrast to the tf-idf matrix), removes some duplicate entries, such as shorter phrases with a lower weight that are included in longer phrases with the same or a higher weight, removes phrases that consist entirely of stop words (done using NLTK's [Loper and Bird, 2002] stop word list[1]), bins phrases of the same weight together, and is sorted by weight, meaning the most important phrases are on top. Again, there is no threshold to exclude phrases, except zero, but since many phrases are removed or combined into more concise phrases (for example the shorter lower-weighted phrases that are removed if they are included in longer higher-weighted phrases), the number of phrases in this dictionary is greatly reduced. Also contributing to that is the fact that in case of Method 1, it does not include the phrases of all the other users, which are zero-weighted for many other users, as is the case in the sparse matrix of the raw tf-idf output, which includes all phrases for all users, in every user column. The reason for binning phrases is merely for human reading convenience, as it makes it simple to read a weight and have all the phrases for that weight next to it. This step is entirely optional.

For all these reasons, this phrase dictionary output is better suited for human readers wanting to get an overview of a user's top phrases, rather than the raw output of the tf-idf calculation or the sequential pattern mining algorithm.

3.1.2. Authorship Attribution

Different methods of authorship attribution are described in Section 2.1.3. To re-iterate, authorship attribution is about finding the author of a text of unknown authorship, in most cases out of a set of candidate authors, given a set of texts of known authorship.

In this case the training phase of the process consists of gathering and weighting the phrases per author as features for the attribution using the known texts. The training data are the texts that are used during the phrase extraction process. The test phase consists of now attempting to assign authorship to "unknown texts". The test corpus of unknown texts are the texts of users that were not used during

[1] NLTK version 3.4.5

phrase extraction. If recalled, not all texts of users were used due to balancing of topic and user lengths. Since the start and end positions of the actually used texts during phrase extraction are saved in meta files, the unused texts can now be used as unknown texts. Since they are already labeled with who the real author is, the accuracy of the authorship attribution using the previously extracted phrases as features can be measured.

The testing phase of authorship attribution is essentially the same for both types of input, which are the outputs of the phrase extraction methods. The only difference is how the input data has to be handled. Since the output of both phrase extraction methods differ slightly in format, they need to be transformed into the same format of phrases and their weights per user. For example, the output of sequential pattern mining algorithms uses support as weights, that is the number of sentences a phrase is contained. These support values need to be normalized, as otherwise authors of longer texts get much larger support values, which would would give them an unfair advantage if not taken care of. This is essentially just a data wrangling task that depends on the implementation.

The "meat" of the attribution process is how the author candidates for a text of unknown authorship are ranked. For each subreddit of the initial subreddit list, that is also for each output of the phrase extraction process and for each user of the output of the phrase extraction process, the user's unused text is used as the text of unknown authorship. Then, there exist 100, or rather n candidate authors out of which only one is the correct author. n is the number of authors that are left over after the exclusion of invalid authors in the phrase extraction process, due to reasons such as too little text, not enough different subreddits (> 1 is necessary for Phrase Extraction Method 2), et cetera. The *score* of a candidate author is calculated as follows: The weights of the phrases of the known candidate author that intersect with the phrases in the text of unknown authorship are multiplied by the n-gram counts in the latter text, and each product is added to an overall sum, which is the score. This results in the most similar authors being ranked high in the output list of ranked candidate authors, which is the list of candidate authors sorted by score, while the most dissimilar ones are ranked low. Ideally, the most similar author, which is also the true author, is ranked at the top position. This also accounts for a possible miss-classification, as even if the wrong author is in the top position of ranked authors, the true author is most likely still in the top range, so the candidate set can be reduced. In this score function, texts that are longer get higher scores, but there is only ever one unknown text for each attribution, and the texts of candidate

```
calculate_score(candidate author ngrams weights,
                unknown text):
  score = 0.0
  for all ngrams in unknown text:
    score += (frequency of ngram in unknown text) ×
             (weight of ngram in candidate author weight list)
  return score
```

Listing 3.1: Simplified score function to rank candidate authors. This function can be used for multiple candidate authors to compare them with an unknown text. Sorting these author/scores tuples afterwards should result in the correct author ranked at the top position.

authors were balanced, so this issue is mitigated in this way. This score function is rather simple but works well for this application, as it does not try to enhance the results in any other artificial way by using for example other meta-data, but relies solely on the phrases and the weights that were calculated during the phrase extraction process. Listing 3.1 shows the score function in a more legible way. The score function was inspired mainly by the *match* function used by Clark and Hannon [2007], which uses a similar but more nuanced computation for a different set of features.

Figure 3.4 shows a diagram of the overall architecture and the process of authorship attribution, and how the phrase extraction fits into the process as the training phase.

3.2. Implementation

The implementation used for this thesis of the thus far described concepts and system follows fairly straightforward according to the given diagrams, in particular Figure 3.1 gives a good overview of the overall pipeline. The actual implementation consists of a few more intermediate steps, mainly about data cleaning and exploration after the data retrieval process, and a few testing phases that try out proposed future steps on contrived data before letting the concepts run on the real data.

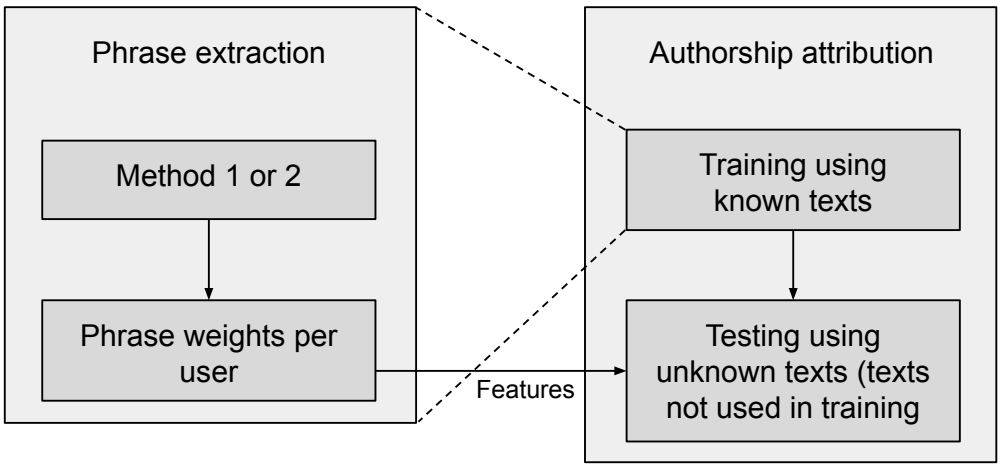

Figure 3.4.: System and process architecture of authorship attribution. The "training" phase of the overall authorship attribution process is made up of the phrase extraction process, which extracts the most important phrases per user from known texts and weights them accordingly. These are then used as features for the "testing" phase and the actual *attribution* phase of the overall authorship attribution process, which is the phase of attributing authorship to texts of unknown authorship, in this case the texts that were not used in the phrase extraction process.

Overall, fifteen consecutive Python[2] Jupyter Notebooks [Kluyver et al., 2016] are used which represent the flow of the pipeline. Figure 3.5 shows all the pipeline steps that are used in the implementation. The following sections describe these steps in detail.

3.2.1. Data Retrieval

The data is retrieved from Reddit via Pushshift [Baumgartner et al., 2020], and its API PSAW[3]. For each subreddit in the initial subreddit list (Section 1.5.3) its top 100 most prolific commenters in the past six months are gathered. For each of those, their last 10,000 comments in that subreddit are retrieved, as well as their own top five subreddits in which they comment. This type of data lends itself well to both phrase extraction methods. The data retrieval process is somewhat of a bottleneck due to the rate limitations imposed on the retriever, and also the amount of data. A bad internet connection as well as the rate limitations make for a slow retrieval process, but leaving the downloads running overnight makes it bearable.

3.2.2. Data Preparation

As the retrieved Reddit data is not plain text, but text marked up via Reddit-specific Markdown, these unneeded symbols need to be removed before any further work can be done on the data. The tool *Redditcleaner*[4] was developed and published specifically for this task. It removes the Reddit-specific Markdown and outputs the cleaned text, which still contains the original capitalization of the characters, as well as punctuation. This is due to the fact that for some tasks, these features might still be needed, and if not, they can be removed in a separate data cleaning task. Listing 3.2 shows raw Reddit text and Listing 3.3 displays its cleaned counterpart.

After running Redditcleaner on the raw Reddit texts, the texts are stored in different formats as different tasks might require different input formats. The ones used in later tasks, that is the phrase extraction and authorship attribution, are an all lowercase format with no punctuation, which is used in Method 1 of phrase

[2]Python version 3.8.3 https://www.python.org
[3]PSAW version 0.0.12 https://github.com/dmarx/psaw
[4]Redditcleaner version 1.1.2 https://github.com/LoLei/redditcleaner

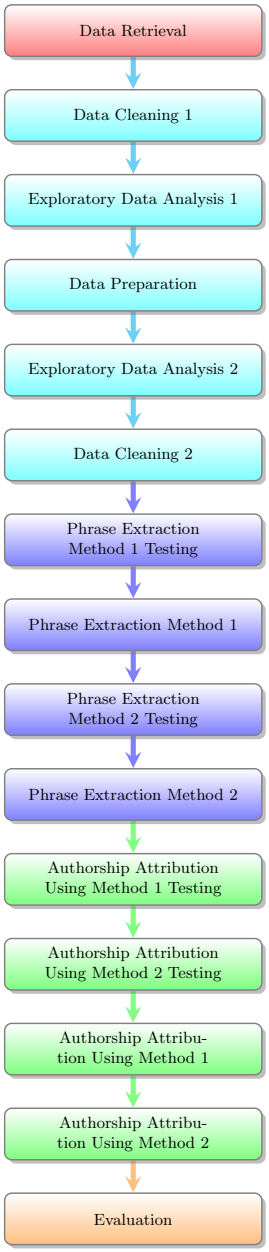

Figure 3.5.: Detailed flow chart of the implemented pipeline. Steps 2 to 6 comprise the data preparation. Steps 7 to 10 concern the phrase extraction, steps 11 to 14 the attribution process, while the last step/notebook is used to evaluate the results.

```
Normal text\n\n**Bold**\n\n*Italic*\n\n[Link](https://fsf.org)\n\n
~~Strike-through~~\n\n`Code`\n\n^(Superscript)
\n\n&gt;!Spoiler!&lt;\n\n# Heading\n\nBullet list:\n\n* Item 1\n*
    Item 2
\n\nNumbered list:\n\n1. Item 1\n2. Item 2\n\n&gt;Quote\n\n
Code block\n\nTable:\n\n|Cell 1.1|Cell 1.2|\n|:-|:-|\n|Cell 2.1|
    Cell 2.2|

\n * Find &amp;#x200B; &gt; "\&gt; the "&gt; hidden\ntext [fsf](
    http://fsf.org)...
This & that in a normal sentence. "manual quote"
```

Listing 3.2: Uncleaned Reddit text in its raw format as it is retrieved from the website. The superfluous characters are due to the Reddit Markdown formatting, which adds text elements like bold font, images, numbered lists, tables, links, et cetera.

```
Normal text Bold Italic
Strike-through Code Superscript
Spoiler Heading Bullet list: Item 1 Item 2
Numbered list: 1. Item 1 2. Item 2 Quote
Code block Table: Cell 1.1 Cell 1.2 Cell 2.1 Cell 2.2

Find the hidden text ... This & that in a normal sentence. "manual
    quote"
```

Listing 3.3: Cleaned Reddit text after Redditcleaner has been run on the raw text shown in Listing 3.2.

extraction, and one with punctuation intact, which is used in Method 2 of phrase extraction, as sequential pattern mining algorithms see sentences as sequences, in which they detect subsequences, meaning the punctuation has to be preserved.

A few data cleaning and data exploration steps are appended in the implementation due to some missing cleaning tasks for some desired formats.

3.2.2.1. Moderator Users

During the initial data retrieval, all obvious bot users, which are not real human users, were removed. During later stages it became apparent that there is also a minority of users that are moderators of subreddits, which means they act as a sort of imposer of rules. This means they often post repeated comments which contain boilerplate terms that indicate why for example another post or comment has been removed. These users have been marked as moderators, but not removed entirely from the process, as they are not always 100% moderator, but also real user, and they provide an easy check if their boilerplate phrases are extracted correctly. However, they do not offer the "real" desired usage of a normal Reddit user. The marking of users that are moderators of a subreddit can be done via Reddit's API, which provides a utility that lists all moderators of a subreddit. PRAW[5], a Python wrapper for this API, was used to achieve this.

3.2.3. Phrase Extraction

For Method 1 of phrase extraction, the tf-idf calculation is done using *scikit-learn*'s[6] [Pedregosa et al., 2011] `TfidfVectorizer`[7], which automatically calculates the tf-idf weights for a list of multiple strings. The arguments used for this function are somewhat important, as they can affect the output vastly. Listing 3.4 shows the exact call to the function using the arguments which have been shown to work best after some research, trial and error. Most important are the `sublinear_tf` argument, which is discussed in Section 3.1.1.1, and perhaps the `token_pattern`, which uses a

[5] PRAW version 6.5.1 https://praw.readthedocs.io
[6] Scikit-learn version 0.23 https://scikit-learn.org
[7] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
min_n_gram = 2
max_n_gram = 5
tfidf_vectorizer=TfidfVectorizer(use_idf=True,
                                 ngram_range=(min_n_gram,
                                     max_n_gram),
                                 token_pattern=r"(?u)\b\w+\b",
                                 norm='l2',
                                 smooth_idf=True,
                                 sublinear_tf=True)
tfidf_vectorizer_vectors=tfidf_vectorizer.fit_transform(
    DF_INPUT_DATA['comments'])
```

Listing 3.4: The arguments used for scikit-learn's `TfidfVectorizer` function. The exact meanings of the parameters and values can be looked up in `TfidfVectorizer`'s documentation[8].

regular expression to capture single-character words as well, which are ignored by default.

.

For Method 2, three sequential pattern mining algorithms to extract phrases are used, to test their capabilities and suitability. The algorithms in question are TKS [Fournier-Viger et al., 2013], which is a *top-k* algorihm, meaning it outputs the top k patterns, BIDE [Wang and Han, 2004], which outputs closed patterns and Gap-BIDE [C. Li and Wang, 2008], which is the same as BIDE, but features gap constraints. See Section 2.2.2.5 for further definitions about the types of sequential pattern mining algorithms. There exist many fast and efficient implementations of these algorithms. SPMF[9] [Fournier-Viger et al., 2016] is used for the first two algorithms mentioned previously. However, since this is a Java library, and the implementation pipeline is made up of Python components, *spmf-py*[10] was developed and published as a Python wrapper for the original Java library. It takes several types of argument and input and formats it to the way SPMF needs it, runs the Java program as a child process, and then retransforms the output back to the Python caller, with a few nice-

[8] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[9] SPMF version 2.43 http://www.philippe-fournier-viger.com/spmf

[10] SPMF-py version 1.3 https://github.com/LoLei/spmf-py

to-have additions, such as as automatic Pandas[11] DataFrame pickling. The latter algorithm is not featured in this Java library however, so a Python implementation[12] was forked and adapted to fit the pipeline. Vanilla BIDE without gap constraints was quickly abandoned in favor of its version with gap constraints.

3.2.4. Authorship Attribution

There are no particular implementation details in the attribution phase of the pipeline that deviate from the concepts described earlier. The score function, as seen in Listing 3.1 is essentially the same for both methods of phrase extraction. Again, there are differences in how the output from the previous phase must be handled, for example the support weights of the sequential pattern mining algorithm need to be normalized, which is done using scikit-learn's `minmax_scale`[13] for min max normalizing, and raw Python for L1 normalization, both of which are tested. If the top phrases dictionary is used, phrases of the same weight need to be unbinned, that is split up from a container of the same weight into multiple tuples.

Three different report files are created after a run of attribution/testing is completed, which means for one subreddit, its 100 users have each acted as the unknown author once, and were compared to all 100 users once:

- *Classification Report* - Created using scikit-learn's `classification_report`[14], it shows an overview of the accuracy values for all classes used in the classification. More on that in Chapter 4.
- *Detailed Results* - For each unknown author, it shows the predicted author, as well as the position in the ranking of the actual true author. Therefore it can be seen how high or low the actual true author was ranked. This is also represented as a percentage value: The position of the true author in the ranking divided by the number of overall candidate authors in the ranking. The overall list is also important because a *confidence value* can be generated, which is the normalized top score decremented by the normalized penultimate score. A small confidence value shows that the scores of the top ranked author and the runner up are very close, meaning there is not much

[11] Pandas version 1.0.5 https://pandas.pydata.org
[12] Pygapbide version 4800bc0 https://code.google.com/archive/p/pygapbide
[13] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html
[14] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

	precision	recall	f1-score	support
A_Ron24	1.000000	1.000000	1.000000	1.000000
A**********[15]	0.500000	1.000000	0.666667	1.000000
Agent_Ayru	0.000000	0.000000	0.000000	1.000000
Altar_Falter	1.000000	1.000000	1.000000	1.000000
AngelsSaints20	1.000000	1.000000	1.000000	1.000000
...	...	...	...	...
unpopopinx	0.500000	1.000000	0.666667	1.000000
zuluportero	1.000000	1.000000	1.000000	1.000000
accuracy	0.826087	0.826087	0.826087	0.826087
macro avg	0.789032	0.826087	0.798913	92.000000
weighted avg	0.789032	0.826087	0.798913	92.000000

Table 3.1.: An example of a classification report, as produced by scikit-learn's `classification_report`. The left column represents the used classes, in this case authors, with the last three rows being averages. The four columns show the precision, recall, F-measure and the support for each class. The meaning of these values is explained in detail in Chapter 4.

difference between them, and the confidence of the output of the top author and supposed true author is low. Vice versa, a high confidence value means the opposite.

- *Mean Rank* - Per subreddit, the mean ranking of the true author. This is the average of the percentage described above.

Also created is a *scores* file for each unknown author, which lists the detailed actual score values of each candidate author, which are used to calculate some of the entries in the tables/files described before. This is the most detailed view that is available for each unknown author.

Table 3.1 shows an example classification report and Table 3.2 an example of one of the detailed results summary of a run.

[15]Some names may be redacted as they are deemed to vile to be displayed, as Reddit imposes no restrictions on what usernames can be chosen.

unknown	pos	num all	%	predicted	confidence
Vasuki44	15	92	0.16304	seikocp	9.58281e-05
unpopopinx	1	92	0.01086	unpopopinx	0.00019
r2k398	2	92	0.02173	aeroeagleAC	0.00015
Naos210	2	92	0.02173	seikocp	0.00023
JJJ_99	1	92	0.01086	JJJ_99	0.00063
...	...	...	...	...	...
Merci-Cabron	8	92	0.08695	unpopopinx	0.00314
idonthavea	4	92	0.04347	Curia-DD	0.00088
HeyMoon69	1	92	0.01086	HeyMoon69	0.00373
NubAlert	1	92	0.01086	NubAlert	0.00425
LWST9	1	92	0.01086	LWST9	0.01731

Table 3.2.: An example of the more detailed view of the results of one attribution run for multiple unknown authors. The leftmost column shows the unknown author, next the position in which this author was ranked in (correct prediction if this value is 1), the number of all candidate authors, the percentage of the ranking (that is the ranking position divided by the number of candidate authors), the predicted author (which might be a different one than the actual true unknown author), and the confidence value of the prediction, as described earlier. A few columns are omitted which exist in the actual implementation due to paper size constraints.

3.2.5. Evaluation

The Evaluation notebook goes through the various result files produced using different configurations in the previous stage and analyses them. Tools used to plot the results are:

- *matplotlib* - General Python plotting[16] [Hunter, 2007]
- *seaborn* - Python visualization, based on matplotlib[17]

All configurations used in the previous stages are compared and plotted, to get a sense of where the best parameters lie, as well as see correlations between differences in the natures of subreddits, the data, and the results. This can be seen in the next chapter.

3.2.6. General Underlying System Details

Regarding the infrastructure on which this implementation was developed, it uses a 6 core CPU at 3.7GHz with 16GB RAM, on the Arch Linux operating system (Kernel 5.7.7). Due to the bleeding edge rolling release cycle nature of Arch Linux, all the libraries and tools that were used utilized their newest versions, but no unstable releases could be detected. All the used version numbers are mentioned in the footnotes in the previous sections. Since vanilla Python is a single-threaded programming language, the possible CPU cores do not matter in this implementation. However, theoretically, multi-threading or multi-processing can be used when implementing the system, as results within one stage of the process do not depend on each other and computations can be run in parallel. The memory limitations do matter, as large matrices such as the raw tf-idf matrix for large inputs may need to be split up and the parts processed on after another instead of holding it all in memory at once, which has been done for a few subreddits.

[16] Matplotlib version 3.2.0 https://matplotlib.org
[17] Seaborn version 0.10.1 https://seaborn.pydata.org

4. Evaluation

The used approaches to retrieve author-specific phrases shall be evaluated on their accuracy if used as features for authorship attribution. In particular the two methods of phrase extraction shall be compared, as well as different configuration parameters within each method. This chapter makes these comparisons, describes used parameters and the data set, shows detailed results, as well as discusses them.

4.1. Evaluation Methodology

The main part of the evaluation of the methods happens right at the end of the attribution step of the pipeline. Since this step is also the "testing" phase of authorship attribution, it produces accuracy results that stem from predicted classes versus actual classes, in this case predicted authors versus the true unknown authors. The result tables/files that are produced at the end of the attribution are described in detail in Section 3.2.4, where example output files that are used for the evaluation can be seen as well. The most interesting output value is the average accuracy of a subreddit within a specific configuration. This means every retrieved user in a subreddit acted as the unknown author once, and was compared to all candidate authors in that same subreddit, 100 authors at most. For each unknown author, the predicted author is saved. The average accuracy of that subreddit within that configuration is then the mean of the correctness of these predictions. For example, if the average accuracy of a subreddit is 50%, it means every other author, or half of all authors, have been predicted correctly based on each unknown author's previously extracted phrases. For a value of 100%, all authors have been predicted correctly, and vice versa for 0%, no author has been predicted correctly. The actual results are described in more detail below. A baseline of for instance 50% is actually very desirable and far better than what an artificial random classifier could achieve. If a random author is chosen out of 100, or n authors, the probability that this

author is the correct one is $\frac{1}{n}$. For an entire subreddit then, the average would also be:

$$\frac{\left(\frac{1}{n} \cdot n\right)}{n} = \frac{1}{n}$$

A more appropriate baseline for comparisons would be another authorship attribution method from the current state-of-the-art. A fitting one is the method of authorship attribution done by Ruder et al. [2016], who use Convolutional Neural Networks (CNN) to apply authorship attribution on large-scale data sets. Their paper is particularly fitting as they evaluate their model (among others) on a Reddit data set of 2 million comments, which is essentially the same data set which is used in this thesis, only limited to the most prolific commenters of the /r/gaming subreddit, which is used here as well, and retrieved from a different temporal range. They also test other methods on this same data set, therefore these results can be used for comparisons as well. These results are listed in Table 4.1 for reference here, and discussed in Section 4.5.7.

4.2. Parameters

Different configuration parameters for each phrase extraction run within each method have been chosen, so that their suitability can be evaluated.

For phrase extraction Method 1, these parameters are:

- *Full tf-idf matrix (raw)* - The raw output of the tf-idf calculation, all phrases for all users and their weights.
- *Full tf-idf matrix (no stop word phrases)* - The same as the above but phrases that consist of only stop words are removed.
- *Top phrase dictionary for each user* - The post-processed list of top phrases for each user, as described in Section 3.1.1.3.

Also, two attribution runs have been done for all of these in which the unknown text stems either from the unused text in the same subreddit from where the users were retrieved and which was used for the extraction of phrases, or from text of

Model	Reddit		Average
Number of authors	10	50	
SVM+Stems [Allison and Guthrie, 2008]	35.1	21.2	60.0
SCAP [Frantzeskou et al., 2007]	46.5	30.3	65.3
Imposters [Koppel et al., 2011]	32.1	16.3	43.6
LDAH-S [Seroussi et al., 2011]	43.0	14.2	49.9
CNN-char [Ruder et al., 2016]	58.8	37.2	73.4

Table 4.1.: Different state-of-the-art models and their F_1-scores (in %) on the Reddit /r/gaming data set, as reported by Ruder et al. [2016]. The rightmost column shows the results of each model averaged over different domains, while the Reddit column focuses on the Reddit /r/gaming data set exclusively. Also worth noting is that the performance for the Reddit data set went down even further when more authors were used, decreasing from the results seen here to the range of 20% to 10% when author numbers from 200 to 1,000 were used.

the other five subreddits that have been retrieved per user. Using either will show whether the phrases also hold up as features outside of the initial topic.

This means, overall, for Method 1 there are six configurations that are used for each subreddit.

For phrase extraction Method 2, the parameters used are:

- *Phrase input type* - Raw sequential pattern mining output or top phrases dictionary
- *Algorithm* - The used sequential pattern mining algorithm. (TKS or Gap-BIDE.)
- *Normalization method* - The method of normalizing the support weights of the sequential pattern mining algorithm to a range of 0 to 1. (Either min max or L1 normalization, as described in Section 3.2.4.) This is a configuration step that can also happen during the attribution phase, if the weights are not normalized as part of the feature extraction phase.

As Method 2 of phrase extraction retrieves phrases from multiple subreddits per user in any case, no additional configurations regarding subreddits are made. The unknown text is always the unused text from a user's five subreddits.

Combining these configuration parameters makes for eight different setups for Method 2, which are evaluated for the attribution process.

4.3. Data Set

The choice for Reddit as a source of text data is described in Section 1.5. Here, the specifics of the retrieved and used data set are described.

The initial list of subreddits to be used is the following:

- *AmItheAsshole* - People judging other people based on stories they post
- *askreddit* - General questions for the entire user base
- *books* - Discussion about books
- *boxoffice* - Discussion about the business of movies
- *classicwow* - About a classic version of World of Warcraft
- *games* - Content about games
- *gaming* - Discussion about gaming

- *HomeworkHelp* - Users helping others with their homework
- *MakeNewFriendsHere* - Connecting strangers for friendship
- *movies* - General movie discussion
- *news* - General news, mostly US
- *nextfuckinglevel* - Content about extreme situations
- *politics* - General politics
- *teenagers* - Populated by teenagers
- *tifu* - "Today I fucked up", stories about mishaps
- *todayilearned* - Users sharing what they learned today
- *unpopularopinion* - Users sharing their unpopular opinions
- *worldnews* - Worldwide news

These subreddits have been chosen for both the fact that they have a high number of users, posts and comments, as well as they differ by topic, user base, and nature, either they pertain to a specific topic, or are more general discussion based.

Again, for each of these subreddits, its most prolific 100 users from the past six months are calculated (for /r/AskReddit only one month was used for this calculation as it has by far the most comments and commenters), and their last at most 10,000 comments on that subreddit are retrieved, as well as comments in five additional subreddits for each user, their other subreddits in which they most contribute.

The following tables and figures show the amount of data and various statistics about the data that was retrieved. Table 4.2 shows the overall statistics for the retrieved data set. Figure 4.1 shows the median comment sizes per subreddit. Figure 4.2 shows the number of comments the top user of a subreddit posted. Figure 4.3 shows the number of commenters that posted in a subreddit's used past period. Figure 4.4 shows the number of comments per subreddit only containing comments of the initial subreddit list, also counting comments that were posted in the used past period of time.

For a detailed view for each subreddit, as well as for the number of actually crawled comments per subreddit, meaning the at most 10,000 last comments per top commenter of these subreddits, see Appendix A. The entire input data set of user comments in a once-cleaned (via Redditcleaner) format can also be accessed online[1].

[1] https://lolei.github.io/msc-dataset

Number of subreddits	18
Number of subreddits after invalidation	15
Number of authors	1,748
Number of comments in the data set	10,642,641
Average comments per author	6,088
Number of comments in subreddit list	5,796,106

Table 4.2.: Overall data set statistics. The number of comments in all subreddits refers to the initial list of subreddits plus the additional top five subreddits per user. The last row refers to the initial list of subreddits only. Not all of the initial subreddits in the list could be used, therefore the final number of subreddits used after the data retrieval shrinks from 18 to 15. See Figure A.2 for the reason.

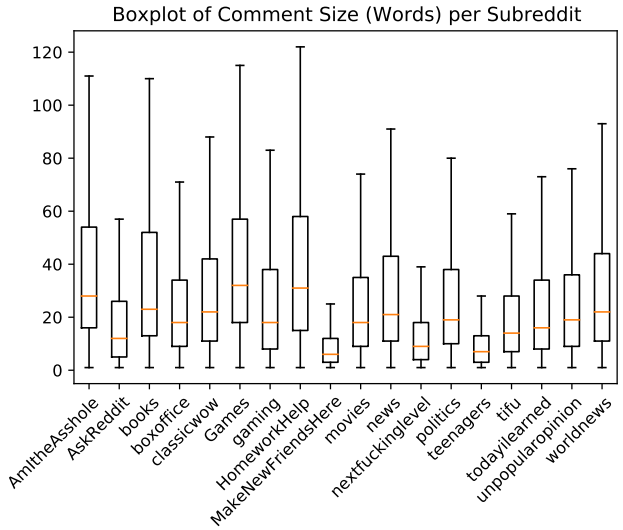

Figure 4.1.: The average comment sizes per subreddit expressed in number of words. As can be seen, they tend to be rather short ranging from fewer than ten words to above 30.

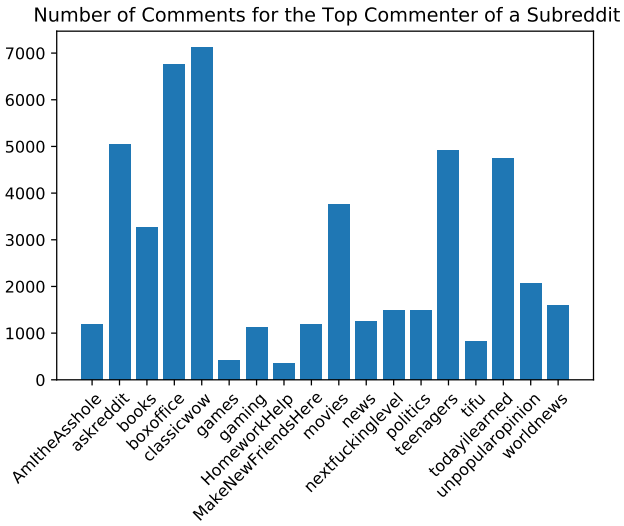

Figure 4.2.: The number of comments the top commenter of a subreddit posted in the last six months (as of creating this data set).

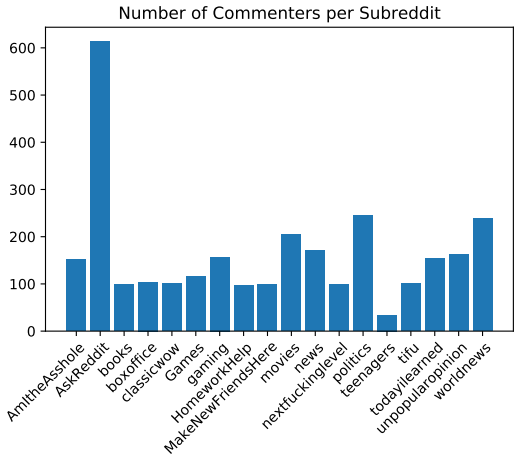

Figure 4.3.: Number of commenters per subreddit. These result from a commenter being a top 100 frequent commenter in a subreddit, or a subreddit being in the top five most frequented subreddits of a mentioned top commenter.

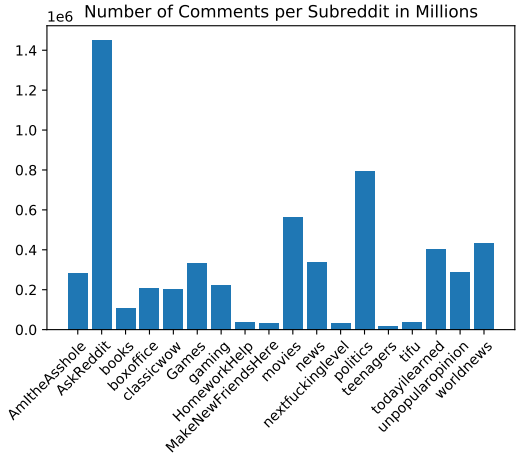

Figure 4.4.: Number of comments per subreddit in millions. This exludes the extra subreddits scraped per user, which are included in the absolute total number of commenters. However, if a subreddit of the initial subreddit list is included in the plus four extra subreddits of a user, it is included here as well. The sum of this plot is the last row in Table 4.2.

4.4. Results

This section shows the results of the phrase extraction and authorship attribution processes for both extraction methods as well as the different configuration parameters described before. Discussion of these results follows in the subsequent section.

First, a few of the result value meanings need to be explained. Classification results can be assessed with different metrics. A common practice is to use *precision*, *recall*, F_1*-score* and possibly *accuracy* values, which are calculated from the same results, but are slightly different, and depending on the use case, one over the other might be used. *Precision* is the ratio of test cases correctly being classified for an author divided by the total number of test cases being classified as written by that author. *Recall* is also the ratio of test cases correctly being classified for an author, but divided by the overall number of possible correct test cases. The F_1*-score*, or F-measure, takes both precision and recall into account, as it is the harmonic mean between the two. *Accuracy* finally is the total number of correct test case predictions divided by the total number of test cases.

The following defines these metrics in a more concise manner:

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

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

$$F_1\text{-}score = \frac{2(Precision \times Recall)}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP means true positive, that is correct positive predictions, similarly *TN* are correct negative predictions, conversely *FP* and *FN* are false predictions. Positive and negative refers to the output of the prediction, in this case a true positive is an author correctly being predicted as the author of a text of unknown authorship, while a false positive is an author incorrectly being predicted as the author of a text of unknown authorship. A true negative refers to the not very visible correct report that a text is not being written by false authors. These values make more intuitive sense in a strict binary classification, but they can be applied here as well.

In any case, the most balanced metric for an overview is the F_1-score. For reference, the results are also reported in accuracy values, as the accuracy is the most "easy" to understand and commonly used performance metric. The following overview tables and plots focus on these two values (in the range from 0 to 1 for 0% and 100%). Detailed results for each configuration, including all metrics can be seen in Appendix B. The following two sections provide the overall results for both methods and all configurations.

4.4.1. Method 1 of Phrase Extraction

These results stem from using the phrases extracted via Method 1, which is the method using the tf-idf calculation. Table 4.3 and Table 4.4 show the average F_1-scores and accuracy of all subreddits for each configuration. This is also visualized in Figure 4.5. These tables and the plot are sorted by the best configuration, that is the configuration that produced the best results. The configuration parameters are explained in Section 4.2. Figure 4.6 shows the average F_1-score for each subreddit

Configuration parameters			Results	
Full/ Top dictionary	**Raw/ No stopword**	**Same/Other subreddits**	**Mean**	**Std Dev**
Full	No stopword	Same	0.961360	0.046247
Full	Raw	Same	0.946004	0.051400
Top		Same	0.919521	0.053162
Full	No stopword	Other	0.817124	0.177068
Full	Raw	Other	0.756692	0.172909
Top		Other	0.730771	0.180843

Table 4.3.: Results for Method 1 expressed in mean F_1-score for all subreddits per configuration (Sorted by best configuration). Evidently using the same subreddit for the unknown text is always better than using different subreddits for the unknown text. Within those two differences, the full tf-idf matrix with no stop word phrases ranks best, then full and raw phrases, finally only the top phrases (that is the "dictionary").

using the best overall configuration, and is sorted by best subreddit. Figure 4.7 does the same but for all configurations. This is the most detailed overview results graphic. Finally, Figure 4.8 shows how many authors were used during the attribution phase, as some of the authors had to be excluded during previous stages due to reasons concerning the amount of text and other issues.

Configuration parameters			Results	
Full/ Top dictionary	**Raw/ No stopword**	**Same/Other subreddits**	**Mean**	**Std Dev**
Full	No stopword	Same	0.969549	0.036381
Full	Raw	Same	0.957204	0.040273
Top		Same	0.938134	0.041549
Full	No stopword	Other	0.846064	0.157143
Full	Raw	Other	0.792917	0.155497
Top		Other	0.775891	0.161149

Table 4.4.: Results for Method 1 expressed in mean accuracy for all subreddits per configuration (Sorted by best configuration). These values differ slightly from the F_1-scores in Table 4.3, but the ordering of best configurations does not change.

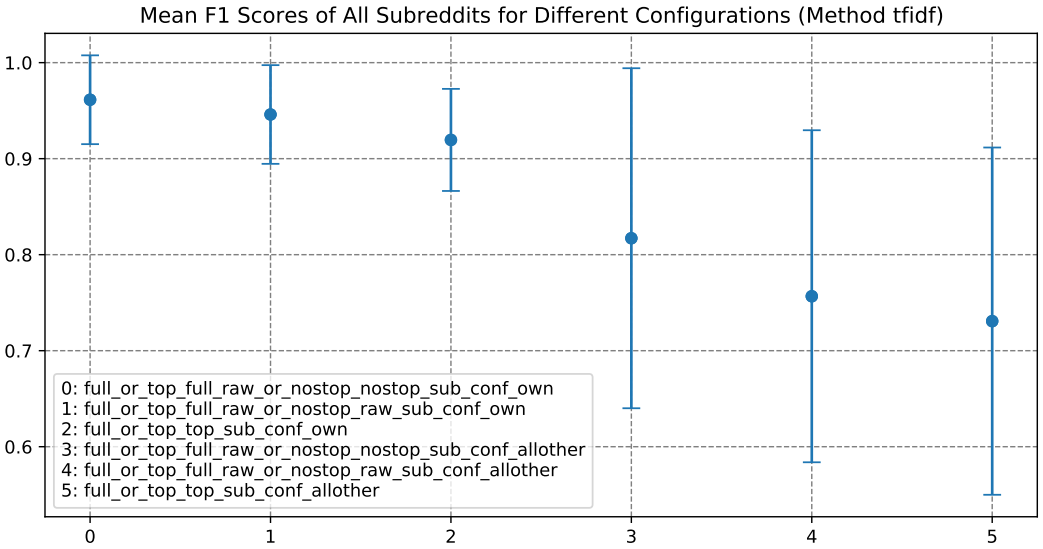

Figure 4.5.: Mean F_1-scores of all subreddits for the different configurations of Method 1. This plot reflects the data of Table 4.3.

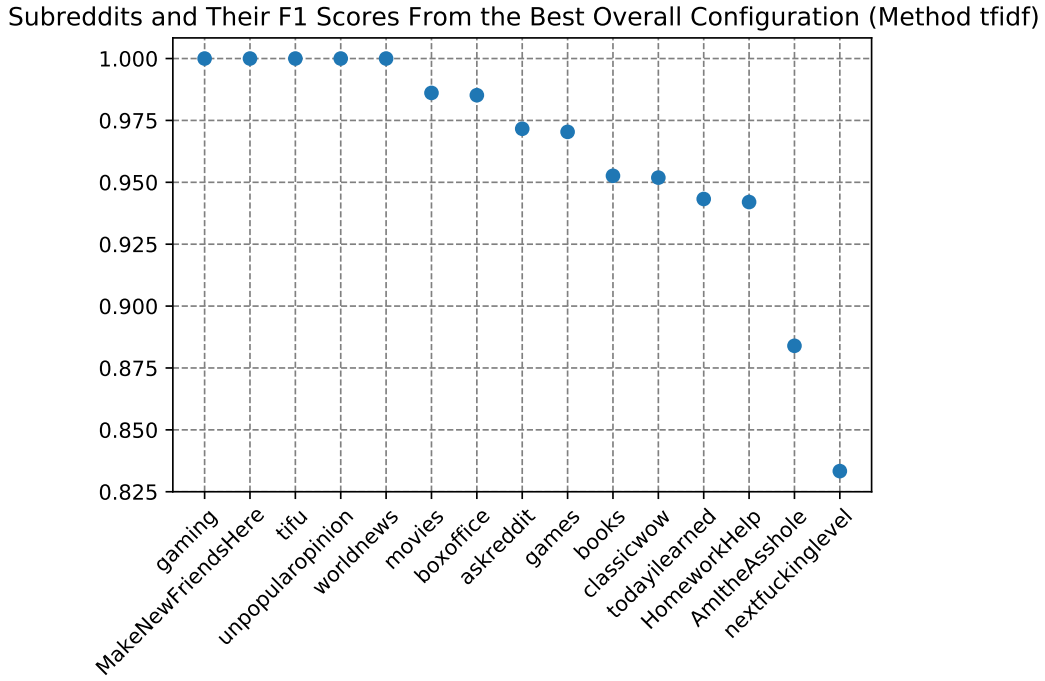

Figure 4.6.: These are the F_1-scores for all subreddits from the overall best configuration of Method 1. The best configuration is the one in the top row in Table 4.3.

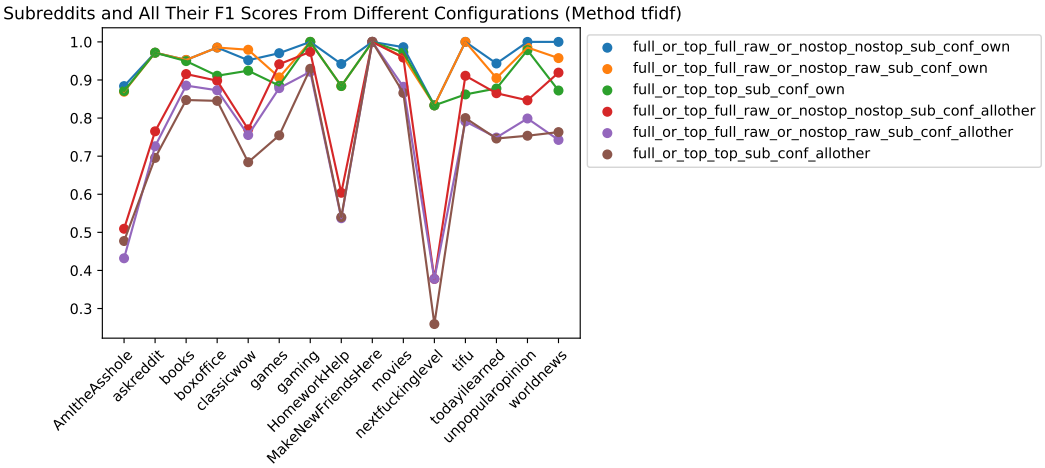

Figure 4.7.: Each subreddit's F_1-scores for each configuration (Method 1).

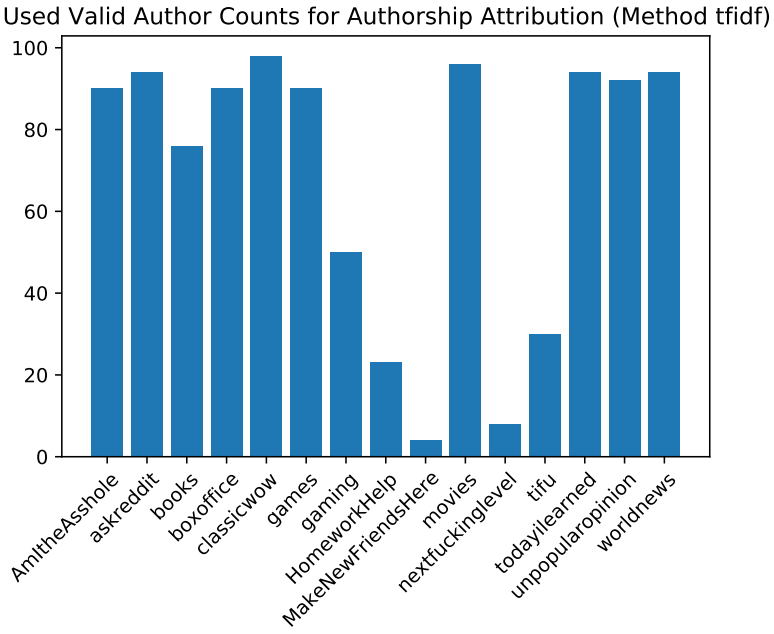

Figure 4.8.: The remnant authors that remained valid and were not excluded during the phrase extraction process with Method 1, due to reasons such as too little text. Most subreddits remained in the 80 to 100 range, while a few had a considerable number of authors removed. This is the number of authors that was used during the attribution process, and from which these results come.

Configuration parameters			Results	
Raw/ Post-processed (Top dictionary)	**Algorithm**	**Normalization**	**Mean**	**Std Dev**
Raw	TKS	L1	0.651988	0.156021
Post	TKS	Min Max	0.442776	0.143869
Post	TKS	L1	0.354701	0.214940
Post	Gap-BIDE	Min Max	0.298410	0.218361
Raw	TKS	Min Max	0.281912	0.153379
Post	Gap-BIDE	L1	0.229138	0.160407
Raw	Gap-BIDE	L1	0.097961	0.154975
Raw	Gap-BIDE	Min Max	0.073465	0.097375

Table 4.5.: Results for Method 2 expressed in mean F_1-score for all subreddits per configuration (Sorted by best configuration). The TKS algorithm is overall better than the Gap-BIDE one. But overall performance is worse than with Method 1. The best configuration is using the raw output of the TKS sequential pattern mining algorithm and normalizing the weights via L1 normalization.

4.4.2. Method 2 of Phrase Extraction

These results come from using the phrases extracted via Method 2, which is the method using sequential pattern mining. Table 4.5 and Table 4.6 show the average F_1-scores and accuracy of all subreddits for each configuration, sorted by best configuration. The same is also visualized in Figure 4.9. Figure 4.10 shows the average F_1-score for each subreddit using the best overall configuration, and is sorted by best subreddit. Figure 4.11 does the same for all configurations. Figure 4.12 shows how many authors were used during the attribution phase, as some of the authors had to be excluded during previous stages due to reasons concerning the amount of text and number of subreddits that could be retrieved per user, as for sequential pattern mining, a number of at least two subreddits, or topics, is necessary.

Configuration parameters			Results	
Raw/ Post-processed (Top dictionary)	**Algorithm**	**Normalization**	**Mean**	**Std Dev**
Raw	TKS	L1	0.696650	0.152929
Post	TKS	Min Max	0.494850	0.147528
Post	TKS	L1	0.392661	0.215710
Post	Gap-BIDE	Min Max	0.344182	0.211250
Raw	TKS	Min Max	0.330966	0.162568
Post	Gap-BIDE	L1	0.269466	0.175496
Raw	Gap-BIDE	L1	0.138488	0.168912
Raw	Gap-BIDE	Min Max	0.112731	0.111791

Table 4.6.: Results for Method 2 expressed in mean accuracy for all subreddits per configuration (Sorted by best configuration). These values differ slightly from the F_1-scores in Table 4.5, but the ordering of best configurations does not change.

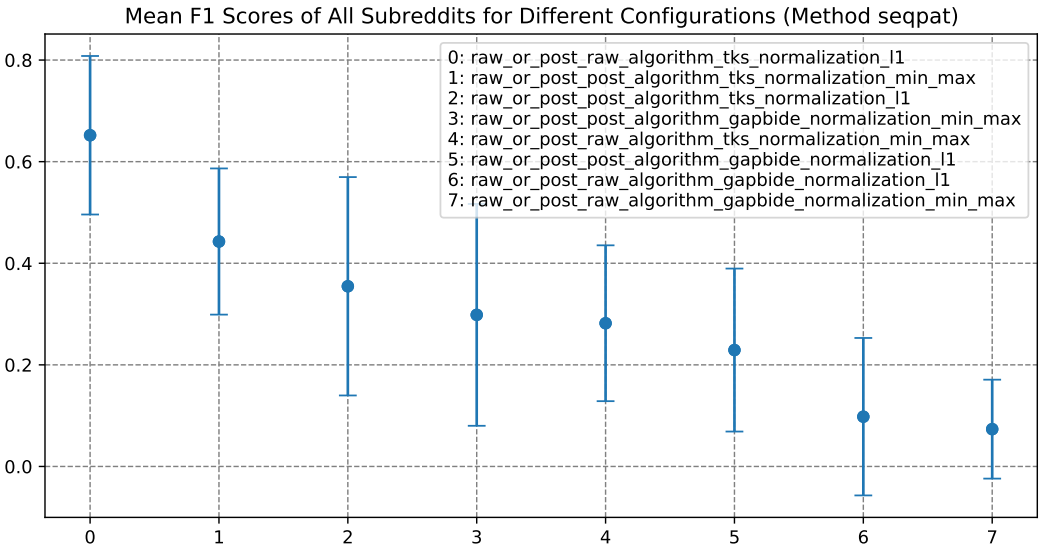

Figure 4.9.: Mean F_1-scores of all subreddits for the different configurations of Method 2. This plot reflects the data of Table 4.5.

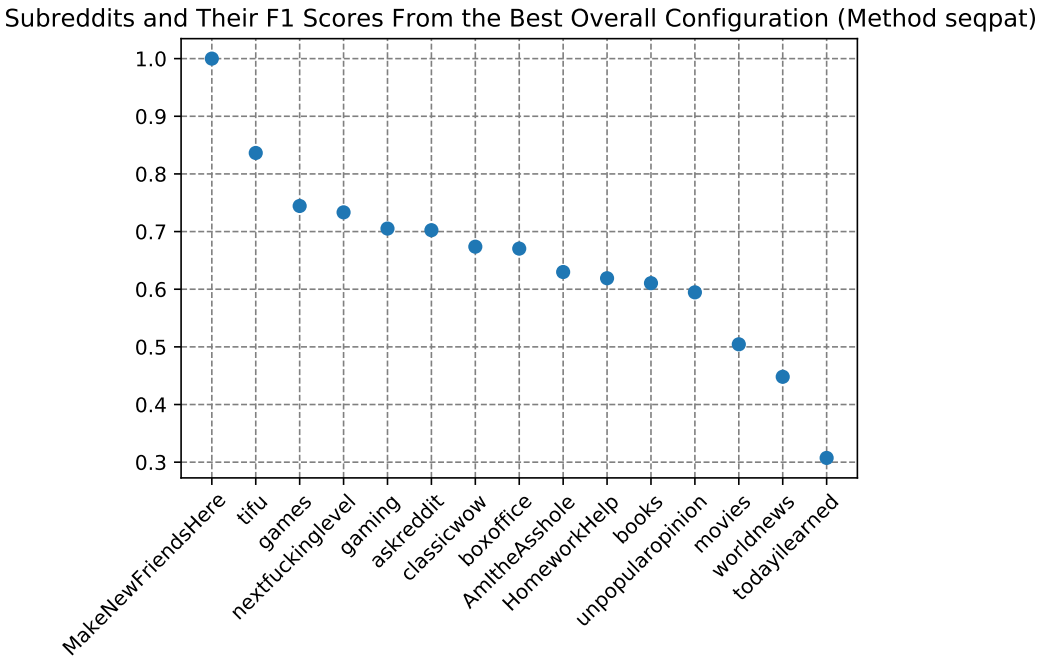

Figure 4.10.: These are the F_1-scores for all subreddits from the overall best configuration of Method 2. The best configuration is the one in the top row in Table 4.5.

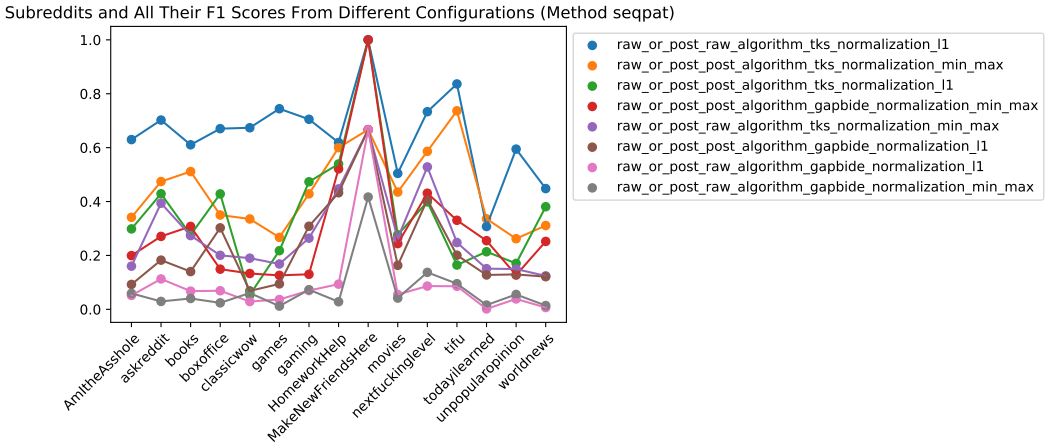

Figure 4.11.: Each subreddit's F_1-scores for each configuration (Method 2).

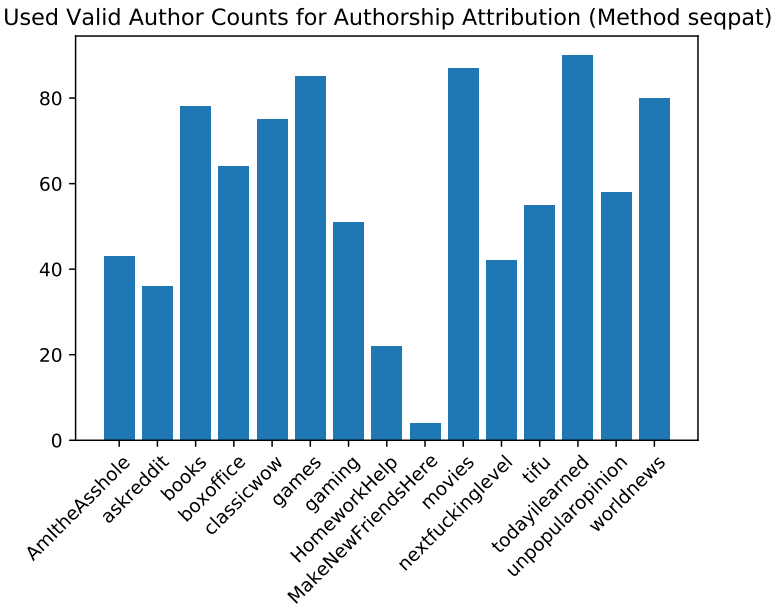

Figure 4.12.: The remnant authors that remained valid and were not excluded during the phrase extraction process with Method 2, due to reasons such as too little text or not having enough (≥ 2) subreddits. More authors had to be removed due to the second constraint than in Method 1. This is the number of authors that was used during the attribution process, and from which these results come.

4.5. Discussion

This section discusses the results listed in the previous section. To summarize, Method 1 of phrase extraction fares better overall than Method 2 when using the extracted phrases as features for authorship attribution. In Method 1, staying within the same topic from which the phrases were extracted works better than venturing outside that topic, indicating the these phrases do not hold up as well over a range of different topics. This may also be the reason why Method 2 works worse overall, as the phrases that are extracted this way already stem from multiple topics, due to the nature of the method. The reason for this may be that some authors post in other subreddits very similar to their main subreddits, which essentially amounts to the same topics, other post in far different subreddits in which they use a different style of writing. This also goes in line with the results of for example the *Common N-Grams* (CNG) approach by Kešelj et al. [2003], which is a similar approach to the one used here, and which also fared worse with generally imbalanced classes in the corpus, as described in Section 1.5.4. However, the imbalanced classes in that work refer to more of the input training data set rather than just using different, but balanced topic in the testing phase. Still, Method 1 achieves relatively good results even outside the initial topic. The mean accuracy scores range from 0.97 to 0.938 within the same topic and from 0.846 to 0.776 with different topics, which is, compared to the baseline discussed in Section 4.1, still admirable. Compared to Method 1, Method 2 fares worse, with mean accuracy scores per configuration ranging from 0.697 to 0.113, although the best configuration in Method 2 still fares fairly well with scores near 0.7. However, the lower range configurations can safely be disregarded for actual use, in particular the Gap-BIDE configurations. Using the TKS sequential pattern mining algorithm generally achieves better results. This may be due to the fact that BIDE is a closed sequential pattern mining algorithm (Section 2.2.2.5), while TKS extracts all k top patterns.

The following discussion is separated into two sections, which discuss either method in more detail, as well as sections that discuss general aspects about the results, methods and other issues.

4.5.1. Method 1

It appears that Method 1 extracts more phrases that contain topic specific words. This is due to the fact that only one topic is used over all users. Ideally, all topic words would be the same for users, and only user-specific topic-agnostic phrases should be extracted, however, many user-specific phrases contain certain words of the topic. These are also reduced if the subreddit itself has no topic, or the topics are very evenly spread. For example in a subreddit like /r/AskReddit, which is more discussion-based, around general topics, there are little to no topic-specific phrases on the top of the list for users, in contrast to a more topic-focused subreddit such as /r/movies.

Still, many completely topic-agnostic phrases are extracted as well. These can be picked out manually if needed for a complete topic-agnostic extraction, using for example the top phrases dictionary. A few examples of these phrases are listed in Section 4.5.8. Also, the sequential pattern mining method is not completely immune to that fault either, but since in that method multiple topics per user are used, topic-specific words are mostly only included if the distribution of topics in the used sentences is uneven in some way.

Regarding Method 1, the output for each user ranges from 100 to 300 phrases, or rather distinct weights, where each weight may contain more than one phrase (in the top phrases dictionary, where phrases of the same weight are binned), but that number tends to be at the lower range of ten. The higher the weight, the more topic specific words are included in phrases, the lower, the more generic the phrases appear, without topic words, but still distinct to the user, although of course less so than the top rated phrases. These more generic phrases are often shared between users, although their weights reflect any differences in usage that is distinct per user. In contrast, with Method 2, the higher phrases tend to be more topic-agnostic or just shorter, while the lower rated phrases get really long (up to the maximum length, if one was specified), and more specific, as they are included in fewer of that users sentences.

Also interesting is that the configuration that removes phrases made up fully of stop words works slightly better for most subreddits than the configuration that includes these phrases. This goes against what Argamon and Levitan [2005] mentioned, as they advise to generally include stop words as features for authorship attribution, as described in Section 2.1.5.1. Still, stop words themselves have been included in

this configuration, only phrases made up *entirely* of stop words have been removed. This might hint at the possibility that this configuration and the overall system might work at topic detection as well, or in fact uses topic detection for the purpose of authorship attribution in this application, even though the overall topic between author comparisons is the same in some runs.

4.5.2. Method 2

It can definitely be seen when manually looking at the extracted phrases that authors use their own stylistic phrases. Some users use some of the same, but those are ranked differently individually per user. Others definitely are user-specific. For example, "My partner and I" and multiple instances of "u" instead of "you" can be seen for two users respectively.

Many single-word "phrases" are on top, because they are of course included in different higher-term phrases. They *could* be removed, but this would in theory also exclude wanted single-word "phrases" such as archaic words. Single-word noun-phrases are also a candidate for exclusion, but ideally, with an equal distribution of topics, these should not be indicative of a user's favored topic. If needed the minimum phrase length an be set to greater than 1, to exclude these words. This has been done at most times in this application, as it turns out there really are not many single-word indicative "phrases" for users.

Intuitive phrases make up about an estimated 25 percent of all gathered phrases after post-processing, that is (full) phrases, that make sense when looking at them, like "that being said". The more "unintuitive" phrases are shorter phrases that are ranked higher than their longer supersequences, such as the single-word phrases which are often at the very top, if they are not excluded.

A limitation of the sequential pattern mining phrase extraction approach is that it needs an even distribution of topic lengths for an author. If an author writes significantly more often about a specific topic rather than others, and all writings are incorporated into the extraction process, topic words and phrases of that topic would be ranked highly. In order to avoid this, and ideally exclude topic words and phrases completely, the topic lengths need to be balanced. For example, if the author writes about five topics, that is subreddits in this case, the topics are balanced and truncated according to the length of the shortest topic. The shortest

topic will be used in full, while a random portion of the length of the shortest topic will be retrieved from the longer topics. This is also a limitation because the data that can be used depends on the length of the shortest topic. On average, about 10% of users had to have their shortest topic removed, because it fell under the limit of a minimum viable topic length. If this happened, the next shortest topic is chosen to serve as the balancing length. This does not matter too much, as as long there are still at least two different topics for an author, the sequential pattern mining phrase extraction approach works. Users who have had only one topic initially crawled or result in only one long enough topic cannot be reliably used with the sequential pattern mining approach, however, it can be easily ascertained in the sequential pattern mining meta information user file, how many topics were used for this user, so their data can be excluded from any next tasks such as authorship attribution if need be. The reason why users on Reddit tend to stick to one subreddit for the majority of their comments is described in Section 2.1.2.4, and an example can be seen in Listing 2.1

As said before, the unused data can be used for different tasks after the Phrase Extraction, for example as test data for a machine learning approach. The data that was used for phrase extraction is marked in a meta information file for each user, which saves the start and end positions of the topics, that resulted from the random selection of length of the shortest topic length.

The weights or ratings used in sequential pattern mining tend to be more intuitive for human readers than those used in the tf-idf weighting scheme. The weights for sequential pattern mining is the *support* of a sequence, or phrase, that is how many other sequences, or sentences in this case, contain this phrase. Given the support, one can easily see how often this phrase has actually been found. In contrast to that, the tf-idf weights are just mostly low numbers close to to 0, as they stem from nothing more than the calculation of the term frequency times the inverse document frequency of a term or phrase. This makes it unintuitive for a human to read at least. For any machines however, this is fine, and also the support weights of the sequential pattern mining results have to be normalized regardless of any further machine-analysis on them is done, such as some machine learning model, which generally need normalized weights of their features, not vastly ranged support values. Also, if multiple known authors are to be used in authorship attribution, those who wrote much longer texts have higher support values for their phrases, even though the ratio of a phrase of their overall phrases might be low. As to not place all authors who write longer texts over authors who write shorter texts these

support values need to be normalized, so the indicative phrase importance weights are reflected appropriately.

4.5.3. Both Methods

In any case, both the Method 1 and Method 2 produce many of the same phrases, if ranked slightly differently, both with different weighting schemes, and different orderings.

In practice, for phrase extraction, either method could be used, there are no obvious benefits or disadvantages of using one method over the other. Only if the extracted phrases are to be used in further steps such as Authorship Attribution, one approach may prove "better" than the other, see Section 4.4 for facts and figures about that. According to the results, Method 1 fares far better in that regard.

The main reason for a choice between the two phrase extraction approaches will stem from the nature of the data set that is being used, as discussed in Section 2.1.2.4. If there are multiple authors with one topic, n-gram tf-idf may be used. If there is one user with multiple topics, sequential pattern mining may be used.

However, depending on the data set and implementation, considerations regarding memory use need to be taken into account. With the retrieved Reddit data set of about 10 million comments the sequential pattern mining using the SPMF implementation is prone to run out of memory a few times in runs of different subreddits, but this can be mitigated. The memory capacity can be increased, and the memory usage decreased, by choosing different algorithm parameters, such as maximum phrase length, et cetera. This may also happen in a different part of the n-gram tf-idf method, but only if a matrix of all users with all their tf-idf weights is to be produced. This step can however be skipped, or the matrix can be split up into multiple parts, with periodic disk writes, to fit it into memory.

The run time is also about the same for each method, with the highest cost coming from the post-processing step, which is optionally applied to both as well, which is discussed in the subsequent section. To speed it up even further, the phrase extraction and the authorship attribution processes can be executed in parallel for multiple users, as the results for one user do not depend on the results of another. Either multi-threading or multi-processing could be used or even the GPU might be

made use of, however, this was not done in this implementation, as the consecutive processing, one user after another, was completed in an acceptable time.

4.5.4. Post-Processing Runtime

The raw results from a sequential pattern mining algorithm include, depending on the algorithm, a lot of redundant and unnecessary information. They include single stop word phrases, phrases made up entire of stop words, many lower-rated subsequences of higher-rated supersequences, et cetera. To get a more human-readable phrase dictionary, some post-processing steps are applied to remove these unnecessary entries, and also bin phrases of the same weight. One step in this post-processing step is the computationally most expensive step in the entire pipeline, which is removing shorter phrases of lower rating, that are included in longer phrases of higher rating, which is in $\mathcal{O}(n^2)$, where n is the number of phrases, as it needs to compare every phrase of lower rating with every phrase of higher rating. The comparison itself is limited to the max phrase length, which is largely limited to about 5, either by the input parameters such as `max_length` present in the n-gram tf-idf approach and some sequential pattern mining algorithms or the inherent property of much longer phrases not being included in a user's top frequent vocabulary, which also excludes them from the output of sequential pattern mining based on the minimum support parameter.

4.5.5. Accuracy and Rank Position

The accuracy of authorship attribution using these extracted phrases can be seen in Section 4.4 using Precision, Recall and F_1-scores. However, the result of authorship attribution is not a binary thing. Each candidate author is assigned a confidence score, and the one with the highest score is the most likely candidate for the unknown text. However, if there are many authors, even having the actual author in a top percentile of the ordered ranked authors is very desirable, as it limits the search space for potential additional investigation. Hence another interesting metric is the position in which the actual unknown author was ranked, and the number of overall candidate authors for this ranking. Precision, recall and F_1-scores do not reflect this property. This confidence score and ranking position is similar to what is described in Section 2.1.5.2.

4.5.6. Correlations and Outliers

There are a few results that may be seen as outliers. In particular, the subreddits /r/gaming, /r/MakeNewFriendsHere, /r/tifu, /r/unpopularopinion, and /r/worldnews have F_1-scores of 1.0 in Method 1's best configuration. On the other hand, the subreddit /r/nextfuckinglevel achieves the worst scores on most configurations in Method 1, but still not "bad" scores for its top two configurations. As for the reason, if Figure 4.7 and Figure 4.8 are compared, there are no obvious correlations regarding author count and results, meaning that the method does not fare worse or better according to the number of authors used, apart from the fact that the results are appear more "extreme" if less authors are used, and do not reflect the average of a subreddit that well. The reason for the results for each subreddit must then depend on the nature of the text that is being written in each, the vocabulary and style of writing that is used, the other topics that the authors frequent in addition to the main subreddit used, and the personality of users that visit these subreddits. It is very hard to ascertain what these factors might be, as they may be hidden deep within a user's personality and the topic which aggregates these types of users. The one subreddit that achieves consistently higher scores for both methods and all configurations is /r/MakeNewFriendsHere, which is also the subreddit that has the lowest number of users that were used in the end. However, these high scores may also be the result of the more natural, enthusiastic and diverse nature of the text that is being used in such a topic. Examples of this can be seen by looking at the user profiles of this subreddit's most prolific authors.

4.5.7. Comparison to State-Of-The-Art

Table 4.1 lists results from different state-of-the-art authorship attribution models on a similar data set such as the one that has been used here, in specific a data set of 2 million comments of the most prolific authors of the /r/gaming subreddit. Using these scores as a baseline, as can be seen, Method 1 outperforms all of the approaches in the Reddit domain, in the /r/gaming subreddit, as well as for some of the other subreddits, Method 2 not so much. For the /r/gaming subreddit, about the same number of authors are used here and in the state-of-the-art models as well, around 50. This makes for a good comparison. However, it needs to be mentioned that the other state-of-the-art models work better in different domains such as email, blogs, IMDb, and Twitter, with Reddit being their worst-performing domain. This

index	weight	longest phrases
0	0.03369	[do you mean nah]
2	0.027102	[need to]
3	0.026926	[sounds like]
4	0.026103	[it sound like]
6	0.025597	[you cant]
...	...	...
309	0.003178	[you have no reason to, you need to learn to, talk to her about it, i dont think its a, have the right to be, to be a part of]
...	...	...

Table 4.7.: This is part of a user's top phrase dictionary resulting from Method 1 (Subreddit /r/AmItheAsshole). As can be seen in the index column, a few rows have been removed according to the post-processing of the raw output that is used to create the top phrase dictionary. Some phrases with the same weight are combined into one row.

means they are better performers for general application, or at least in their multiple well-performing domains, rather than being suited for application to Reddit. Ruder et al. [2016] attribute this to the suspicion that platforms such as Twitter and blogs are more about individual self-expression, with more distinct features such as hashtags on Twitter that can be easily picked up by a neural network, while people on Reddit on the other hand tend to conform more to the conventions of writing that are common to a specific subreddit. They also mention that further research could be done to see if people express themselves differently in different subreddits, which has been explored to some extent in this thesis, as at least different subreddits have been tested. Although to find exact correlations, more research is necessary.

4.5.8. Sample Phrases

Here a few of the extracted phrases for different users are listed, to give an intuition what they actually look like. Tables 4.7, 4.8, 4.9, 4.10, and 4.11, show some of the phrases for different users resulting from Method 1, Tables 4.12 and 4.13 those resulting from Method 2. The brackets indicate multiple phrases with the same weight.

index	weight	longest phrases
0	0.015515	[may wanna]
1	0.014502	[you may wanna]
...	...	...
3	0.012109	[your opinion is]
4	0.011663	[..., your opinion is wrong, ...]
5	0.010649	[..., god i love, ...]
...	...	...
19	0.008196	[i loved that, ...]
...	...	...
22	0.007638	[oh man]
...	...	...

Table 4.8.: Part of a user's top phrase dictionary resulting from Method 1. A few of the more intuitive phrases have been highlighted. Subreddit /r/books.

index	weight	longest phrases
0	0.014816	[no ones saying]
...	...	...
4	0.012524	[imagine actually believing that, ...]
...	...	...

Table 4.9.: Part of a user's top phrase dictionary resulting from Method 1. A few of the more intuitive phrases have been highlighted. Subreddit /r/classicwow.

index	weight	longest phrases
0	0.014602	[u are]
1	0.014365	[u can]
2	0.014212	[u have]
3	0.013986	[u will, u cant]
4	0.013378	[if u]
...	...	...

Table 4.10.: Part of a user's top phrase dictionary resulting from Method 1. A few of the more intuitive phrases have been highlighted. Subreddit /r/classicwow.

index	weight	longest phrases
...	...	...
2	0.010935	[too many assholes, ...]
3	0.010025	[people are idiots, i wish that i, ...]
...	...	...
9	0.008088	[..., you can google it, ...]
...	...	...
12	0.007567	[you have a right to, ...]
...	...	...
14	0.007163	[..., im thinking about, think about my, obligated to, ...]
15	0.007108	[its impossible, ...]
...	...	...
17	0.006832	[doesnt mean anything, a couple of hours, maybe you can, ...]
...	...	...
21	0.006553	[..., i wouldnt know, pisses me of, ...]
...	...	...
28	0.006002	[i realized that]
...	...	...
30	0.005987	[that you know, ...]
...	...	...

Table 4.11.: Part of a user's top phrase dictionary resulting from Method 1. A few of the more intuitive phrases have been highlighted. Subreddit /r/AskReddit.

index	support	longest phrases
...	...	...
15	22	[i think, ...]
...	...	...
22	15	[..., i mean, ...]
...	...	...
25	12	[i thought, ...]
26	11	[trying to, i guess, ...]
...	...	...
28	9	[..., at least, ...]
...	...	...
30	7	[..., feels like, ...]
31	6	[it feels like, i dont know, instead of, ...]
32	5	[..., thought it was, ...]
33	4	[..., looking forward, ...]
...	...	...

Table 4.12.: Part of a user's top phrase dictionary resulting from Method 2. A few of the more intuitive phrases have been highlighted. Here, the weight column is replaced by the support column, as that is the "weight" sequential pattern mining algorithms assign to patterns, that is in how many supersequences this subsequence was found.

index	support	longest phrases
...	...	...
22	13	[..., i read, ...]
23	12	[a little, ..., nah]
24	11	[i thought, at least, ...]
25	10	[..., i mean, ...]
27	8	[..., trying to, couple of, ...]
28	7	[i dont know, like this, a couple, kind of, ...]
29	6	[a couple of, i want to, i guess, i hear, i wish, yeah i, gotta, ...]
30	5	[..., i thought it, i disagree, so many, ...]
31	4	[your opinion is wrong, i feel like, feels like, ...]
32	3	[that being said, i thought it was, fuck fuck fuck, dont know if, pretty sure, i wonder if, ...]

Table 4.13.: Part of a user's top phrase dictionary resulting from Method 2. A few of the more intuitive phrases have been highlighted.

5. Conclusions

This chapter summarizes the previous parts of the thesis and reflects upon the hypothesis, the results, and all important aspects regarding this work.

5.1. Reflections

If recalled, the hypothesis posed in the Introduction presumed that different individuals use different words and phrases in their writing according to their personality. Therefore, these words and phrases may be used to identify people based on their style of writing, that is the specific idiosyncratic words and phrases they use.

This work proposed two methods to extract these kinds of words and phrases from the texts of authors. A major caveat is that these two methods only work in corpora which can be controlled for topic. Method 1, which uses n-gram tf-idf calculations, needs only one and the same topic for multiple authors. Method 2, which uses sequential pattern mining algorithms, works only for multiple topics (≥ 2) for one author. Therefore the corpus used for extracting the phrases needs some way in which it can be ascertained to which topics each author texts belongs. The way how the corpus used in this work is controlled for topic is via subreddit, which are sections related to specific topics on the online forum Reddit.

These two methods for phrase extraction tend to retrieve precisely the types of words and phrases that were suggested in the hypothesis, as can be seen in Chapter 4. Intuitively, these phrases should also work then if used as features for authorship attribution, that is finding out which author wrote a text of unknown authorship, based on candidate authors of which known texts are available, from the phrases were extracted. In practice, the two proposed methods worked better and worse than each other, and depending on different configuration parameters. Method 1 fares better than Method 2 in authorship attribution tests in general, with average

accuracy scores ranging from 0.97 to 0.938 within the same topic and from 0.846 to 0.776 when different topics were used for attribution testing. The latter method scores in the range from 0.697 to 0.113, depending on configuration. The worse-faring configurations can safely be disregarded for actual use, but the ones coming near the 0.7 accuracy score range are still viable for practical use.

Which of the two methods for phrase extraction should be used mainly depends on the nature of the input of text. If there are multiple authors, but one topic, Method 1 may be used. If there is one author, and multiple topics, Method 2 may be used. If there is a combination of these, any of the two methods can be used or combined, if kept in mind that Method 1 fares better overall. For example, if there are multiple authors, and multiple topics, specific author/topic selections can be made to conform the input data to one of the two methods.

Since this type of feature for authorship attribution has been shown to work for this type of data, it may be used in addition of the usual common features that are used in other applications of authorship attribution, as discussed in Chapter 2. It is always beneficial to have more options for features, and if selected, weighted correctly and overfitting is prevented, more features in general tend to achieve better results. Exploring new text data sources such as Reddit is, especially in times where a majority of writings and conversations happen in just such media, of obvious value. Given these facts, it can be concluded that this work achieves a definitive contribution to its field.

5.2. Future Work

This section lists some of the possible future work that can be done related to this thesis. Possibilities include but are not limited to extending the methods used here, using different data sets, or using the thus far used data set in different applications.

5.2.1. Advanced Attribution Methods

The actual attribution of authorship to unknown texts used the *score* function. This function is rather simple or naïve, even though it works as intended. A more

advanced approach could be used instead, that uses for example a machine learning approach that trains the model in different ways, rather than the single "training" phase which was used here, that consisted of gathering the phrases as features and then using them in the score function. Even changing up the score function with different parameters or using an entirely different function may be worth looking into.

5.2.2. More Phrase Extraction Changes and Its Implications

Some configuration parameters for both methods of phrase extraction have been tested, which show great differences in their score and performance. There are of course many more possibilities than what has been done so far. Additional ones and their results could be tested. For example, would Method 1 fare better or worse if more or less phrases are used? Thus far, the used data has been balanced, as it should be, but lowering or raising the balancing middle ground could also result in some interesting results.

5.2.3. Traditional Data Sets

The data set used in this work was specifically gathered for this application. It is suitable due to the fact that it can be controlled for topic, and consists of mainly unstructured text that was composed in an online medium by real people. Since the proposed methods work well with this data set, it would be interesting whether the same methods work with different data sets. For example, a few of the more "traditional" authorship attribution data sets could be tested, as listed by Neal et al. [2017]. However, they need to controllable by topic, otherwise the proposed methods cannot work. It depends on how any new data sets are labeled if this is possible. As theorized in the introduction, an unstructured text genre also might fare better than a text genre for which the style of writing is more dictated, such as newspaper articles or books.

5.2.4. Data Set Possibilities

Since the gathered data set is rather large (~10M comments, see Table 4.2), it could be used for different applications that need text data as well. It is saved in multiple formats, the most interesting one perhaps being the once-cleaned (through Redditcleaner) raw Reddit text data. It contains the natural texts of all comments, with Markdown formatting removed, but including intact capitalization, punctuation, emojis and special Reddit jargon such as /s, which indicates sarcasm. The data set in this format can be downloaded online[1]. It would be interesting to see how well other current state-of-the-art authorship attribution models (in addition to the ones discussed in Section 4.5.7) work on this specific full data set. Since it is text data from an unstructured text genre, an online discussion forum, authorship attribution methods that focus on similar data could be tested on the data set.

5.2.5. Application in Topic Classification

As hinted at before, the presented methods might also work for topic classification. If different topic inputs are presented, especially Method 1 of phrase extraction could be used to extract exclusively topic-related phrases. These could then be used to classify text topics, rather than authorship of texts. This is however an entirely different task than what has been done so far, but similar methods could be used.

5.2.6. Subreddit Differences

The individual subreddits achieve better and worse average attribution scores. As discussed before, there seem to be no obvious correlations when looking at the more visible differences such as number of user comparisons and scores. It might be interesting to find out what causes the different scores in subreddits, what makes one subreddit better suited for these methods than another. It could be the vocabulary that is used, the homogeneity of the topics that are discussed within a subreddit, if people venture more outside common style conventions that may be present in certain subreddits, and many more factors, that are hidden more deeply within each subreddit and the nature of its text data.

[1] https://lolei.github.io/msc-dataset

Appendix

Appendix A.

Detailed Numbers of Comments per User per Subreddit

The following two figures, Figure A.1 and Figure A.2, show detailed plots for each subreddit, and how many comments per top user exist in these subreddit, once time-bound, and once for actually crawled comments. The time-bound comments reflect the calculation of the top 100 most prolific users per subreddit in the past six months. For these users, at most 10,000 comments in these subreddits have then been crawled, which is seen in the second figure. These figures do not reflect the additional five subreddits that were crawled per user.

Appendix A. Detailed Numbers of Comments per User per Subreddit

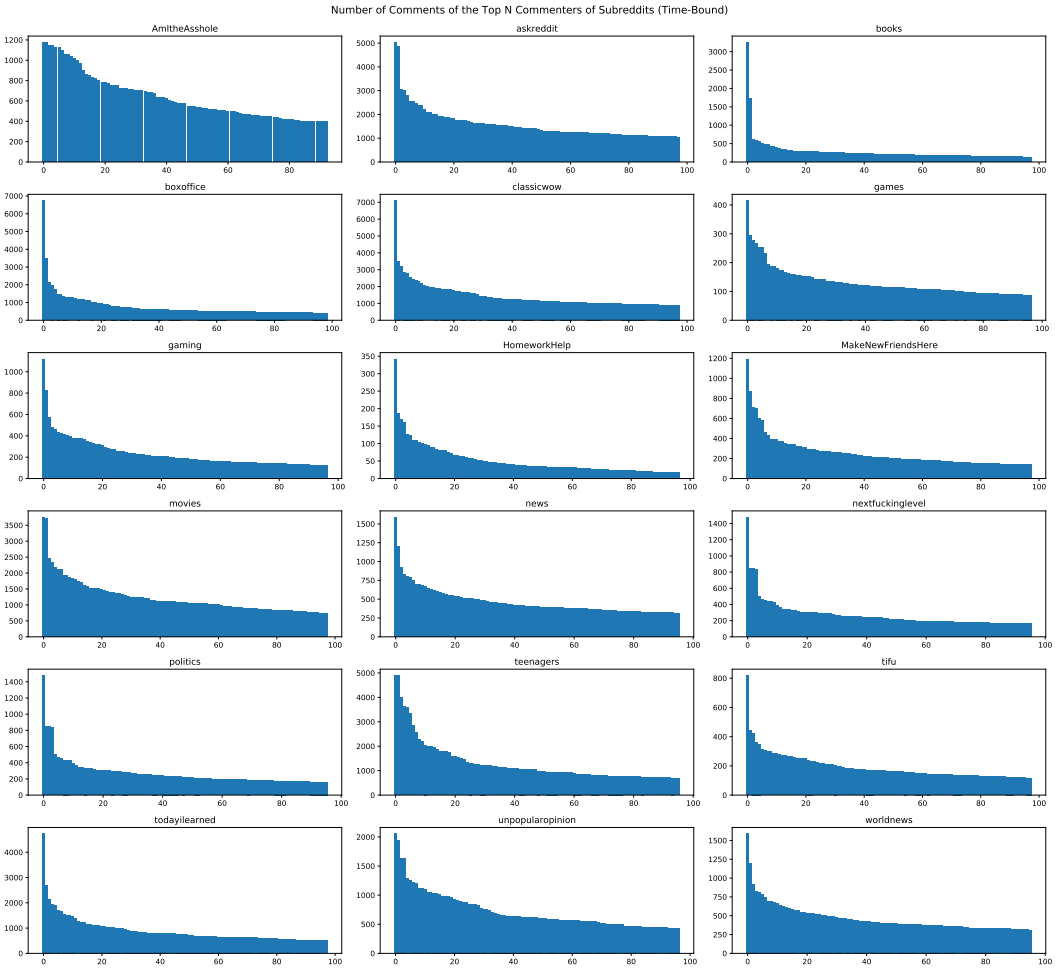

Figure A.1.: The number of comments the top $n \leq 100$ commenters posted in the last six months (/r/AskReddit and /r/politics one month).

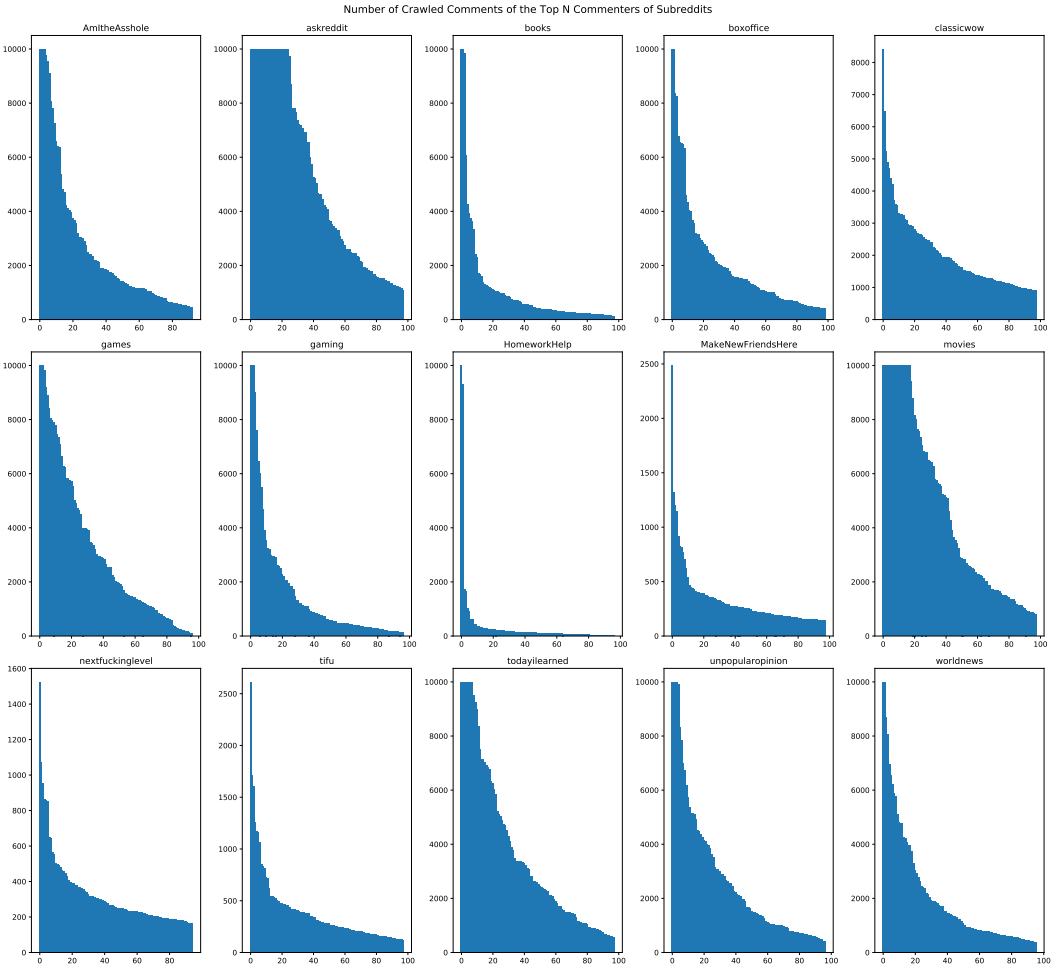

Figure A.2.: The number of actually crawled comments for each top commenter per subreddit. As can be seen, the maximum limit is 10,000 per user, which a few would have exceeded. Also visible in this figure, compared to Figure A.1, is that three subreddits are missing, /r/news, /r/teenagers and /r/politics, as some users got lost during the crawl of these subreddits due to Internet outages. Restarting the download for these subreddits was deemed too arduous and no great loss, therefore these subreddits are disregarded from most future steps in the pipeline.

Appendix B.

Detailed Classification Reports for All Configurations

The following inline figures (Figure B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11, B.12, B.13, and B.14) show all configurations and their results for the /r/AskReddit subreddit. Including all reports here is infeasible due to page count constraints either in printing or when just viewing a digital version, as the total number of reports comes to 210 (15 subreddits $\times$ 6 configurations + 15 subreddits $\times$ 8 configurations). The same report visualizations for all other subreddits can be seen and downloaded online[1]. The /r/AskReddit subreddit was chosen to be displayed here as it is a good "average" subreddit, meaning it produces the most representative results of the overall results. It works well for the best configurations and worse for the lower ranking ones. The used configuration for each classification report is displayed on top of each visualization. The figures are sorted alphabetically by configuration.

[1] https://lolei.github.io/msc-reports

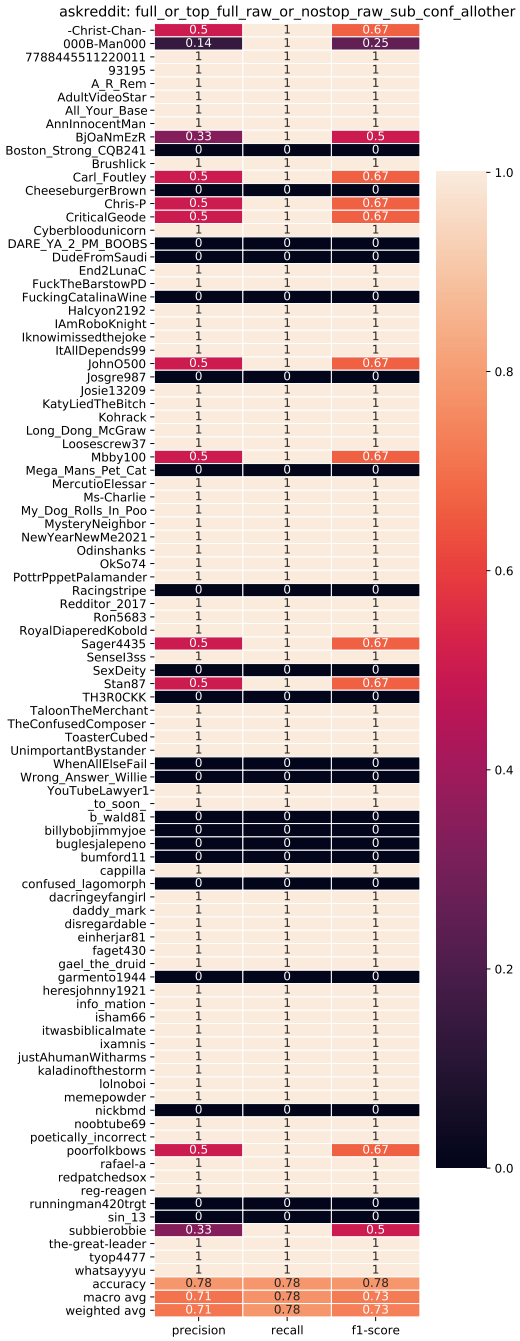

Figure B.1.: Classification report for /r/Askreddit of Method 1.

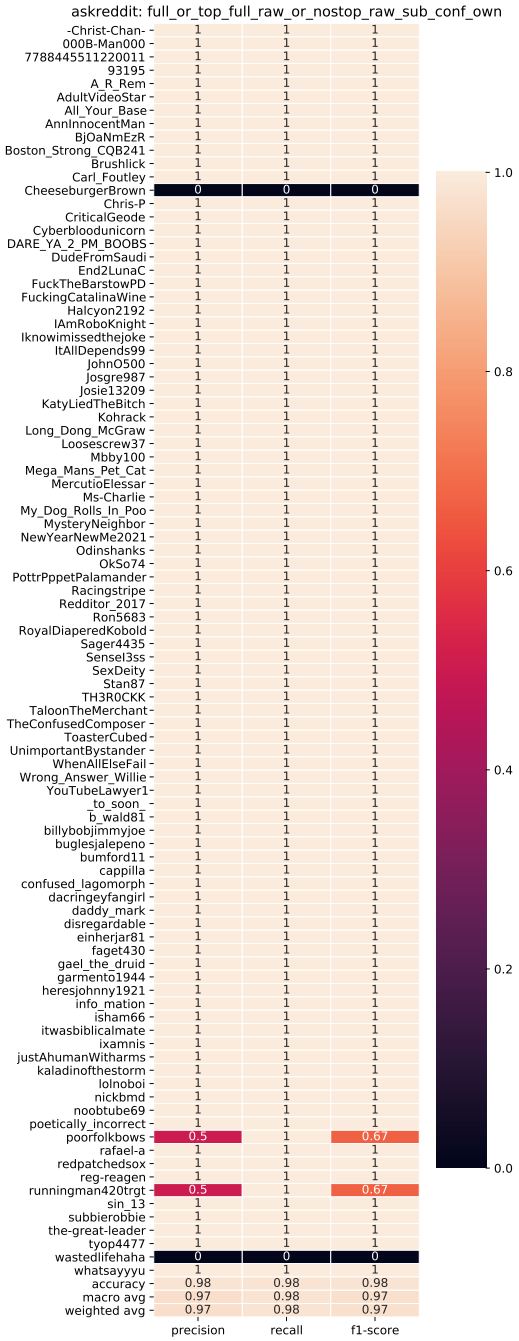

Figure B.2.: Classification report for /r/Askreddit of Method 1.

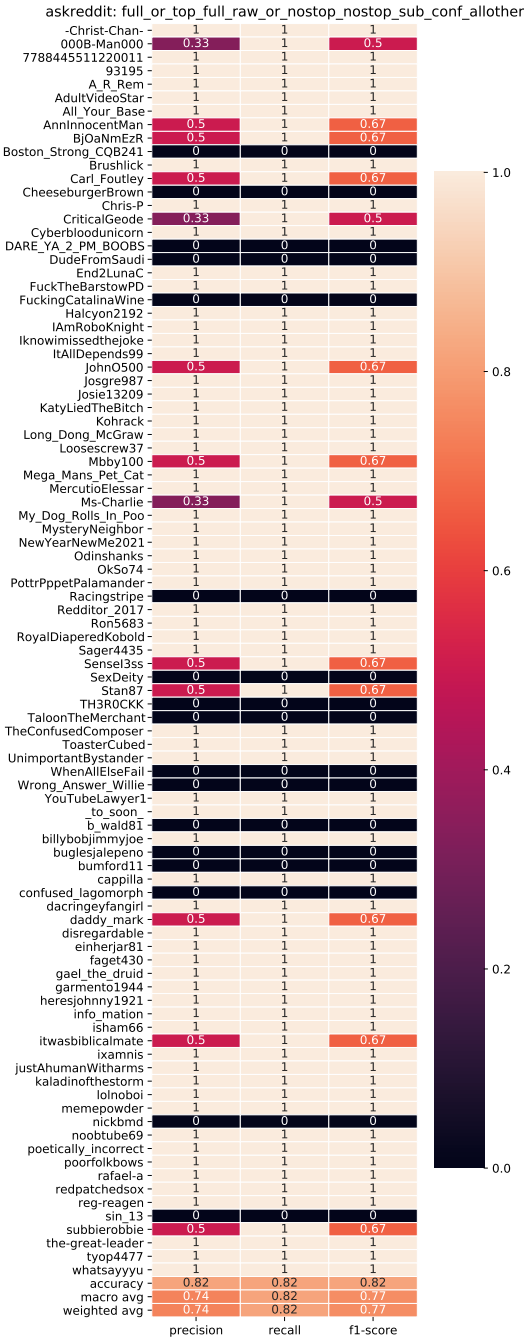

Figure B.3.: Classification report for /r/Askreddit of Method 1.

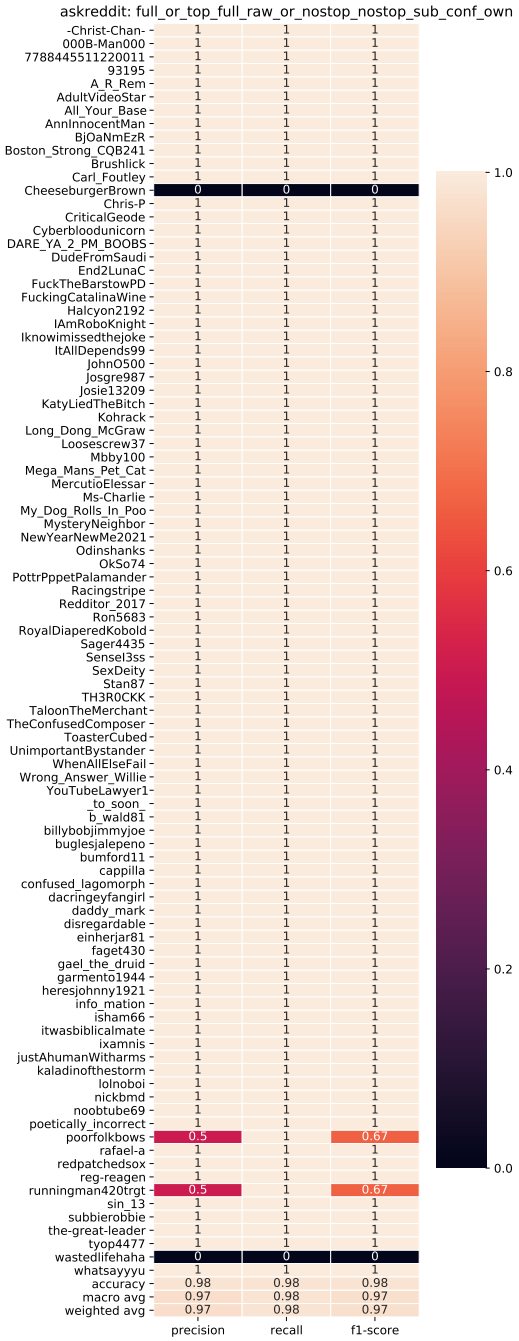

Figure B.4.: Classification report for /r/Askreddit of Method 1.

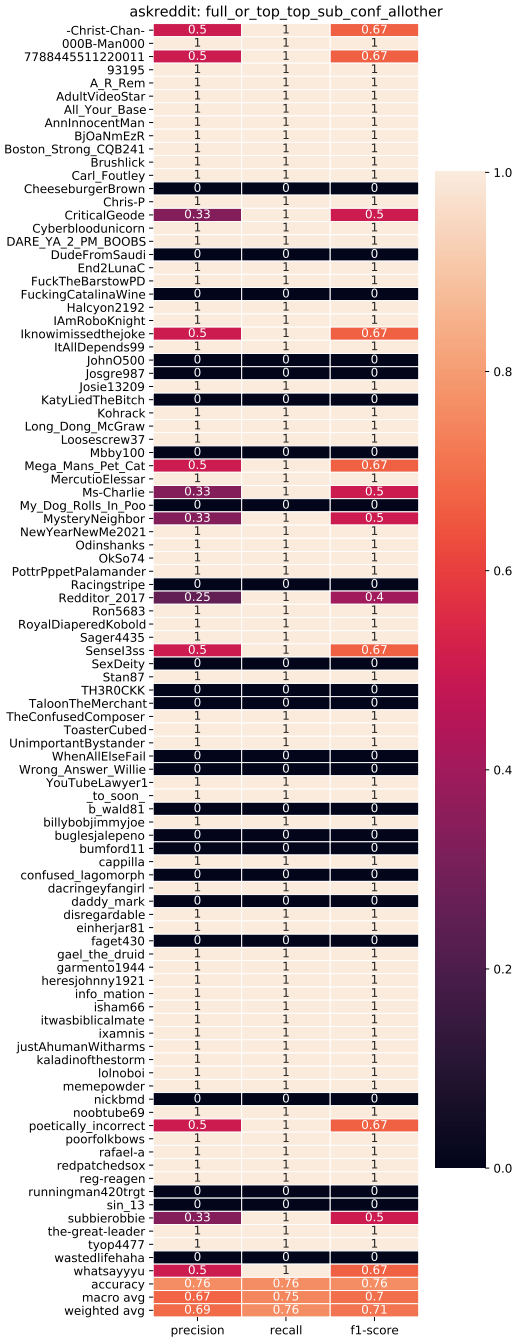

Figure B.5.: Classification report for /r/Askreddit of Method 1.

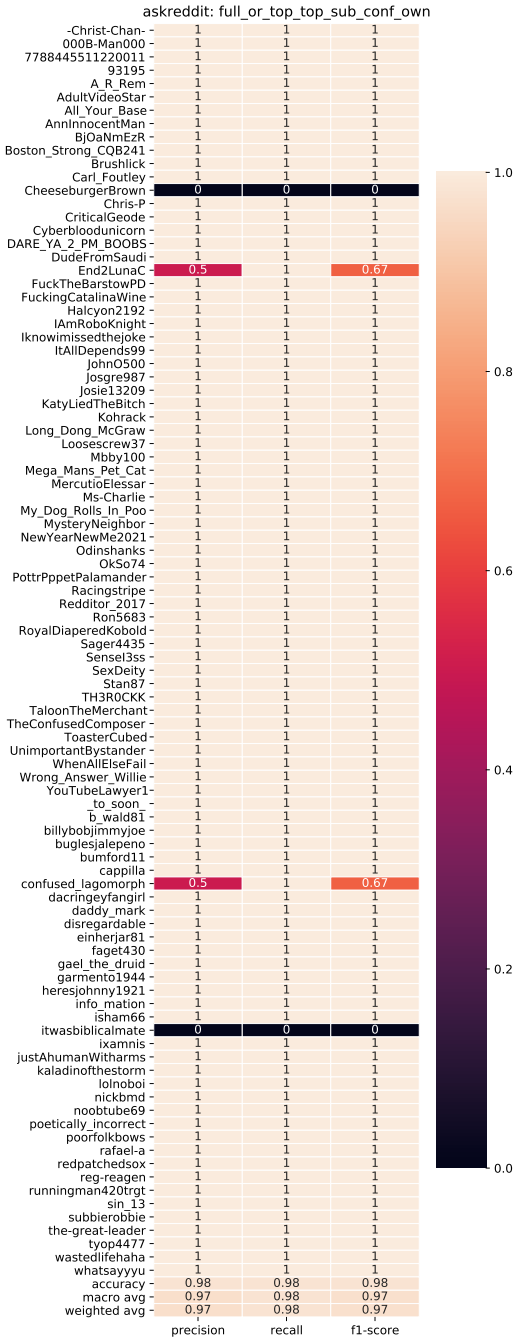

Figure B.6.: Classification report for /r/Askreddit of Method 1.

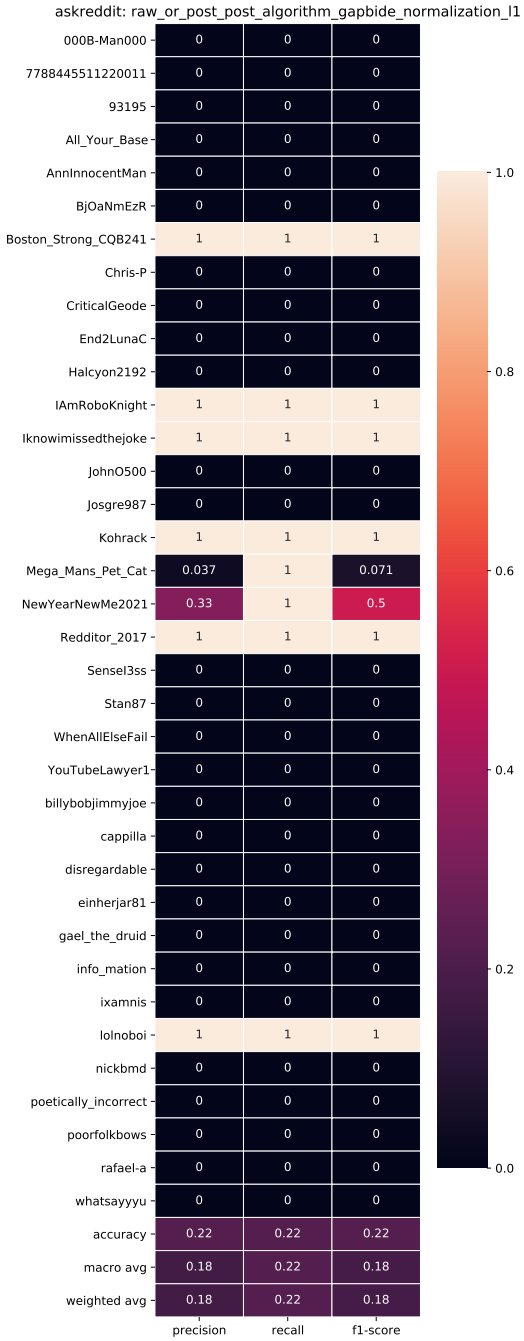

Figure B.7.: Classification report for /r/Askreddit of Method 2.

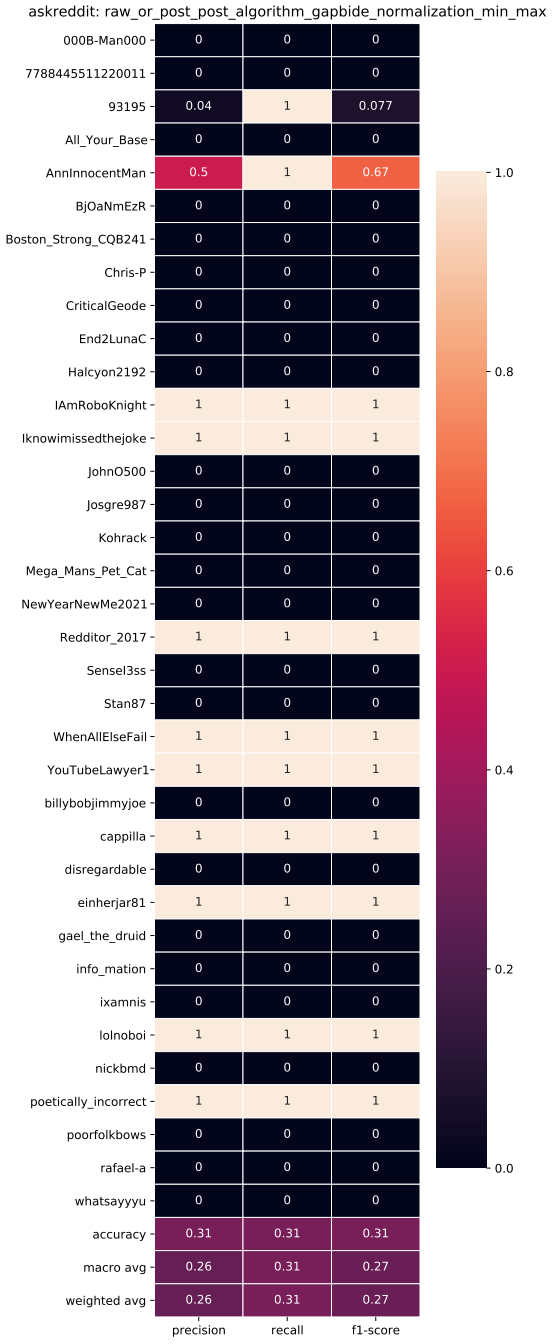

Figure B.8.: Classification report for /r/Askreddit of Method 2.

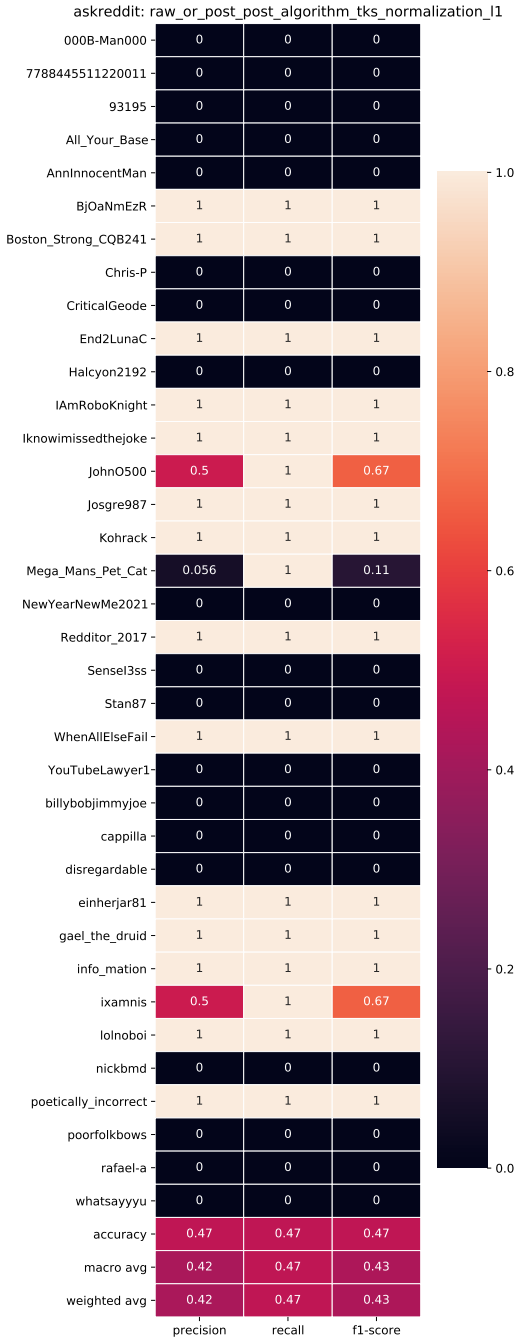

Figure B.9.: Classification report for /r/Askreddit of Method 2.

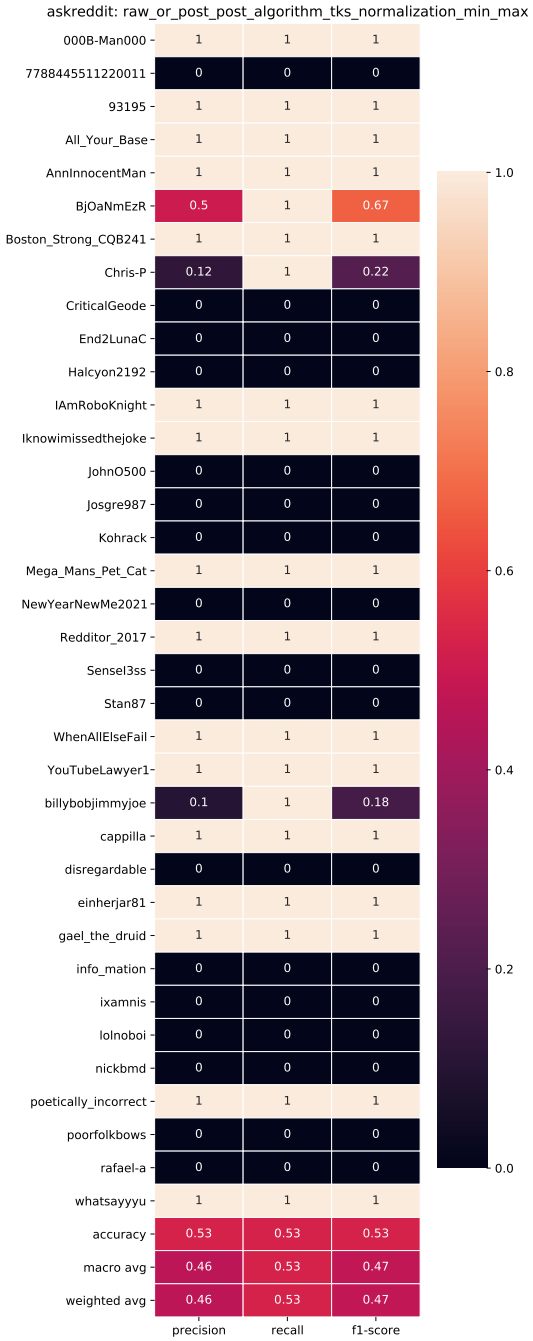

Figure B.10.: Classification report for /r/Askreddit of Method 2.

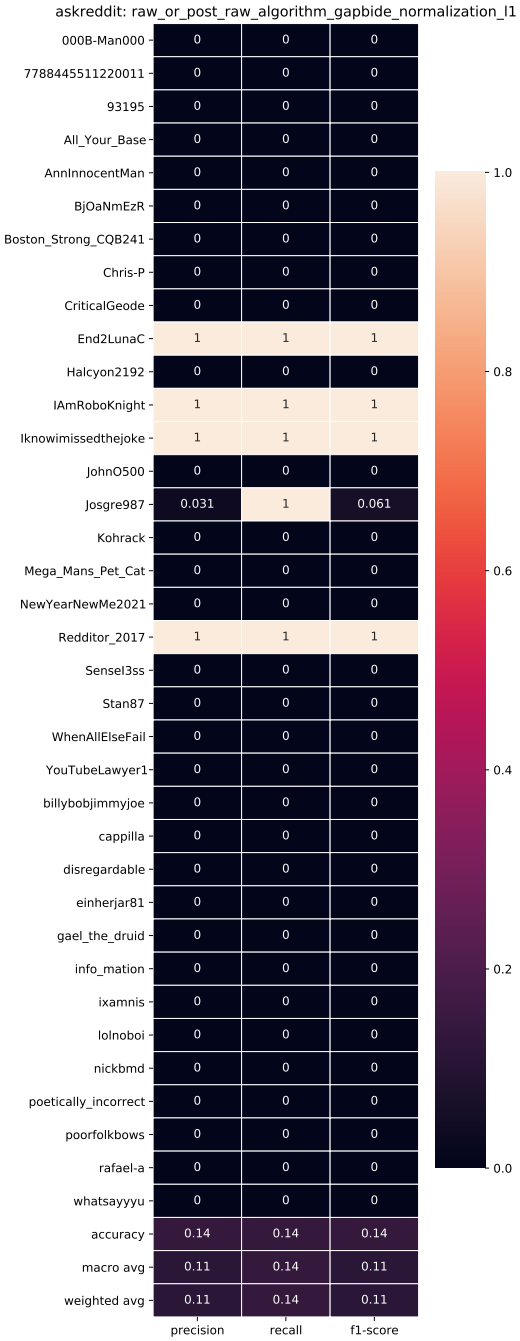

Figure B.11.: Classification report for /r/Askreddit of Method 2.

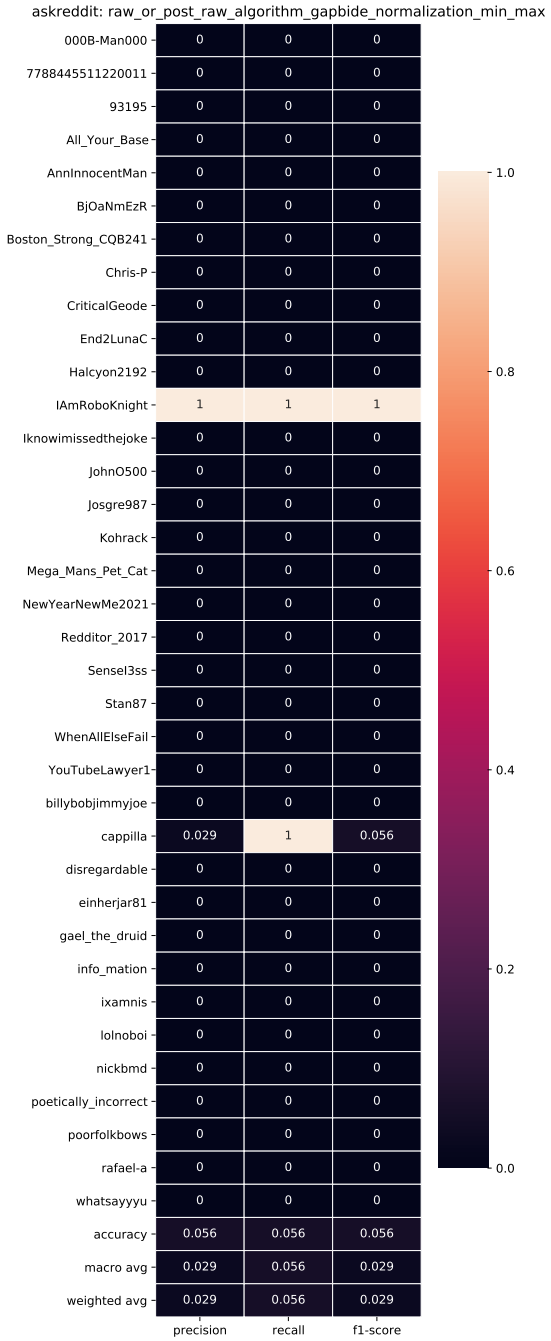

Figure B.12.: Classification report for /r/Askreddit of Method 2.

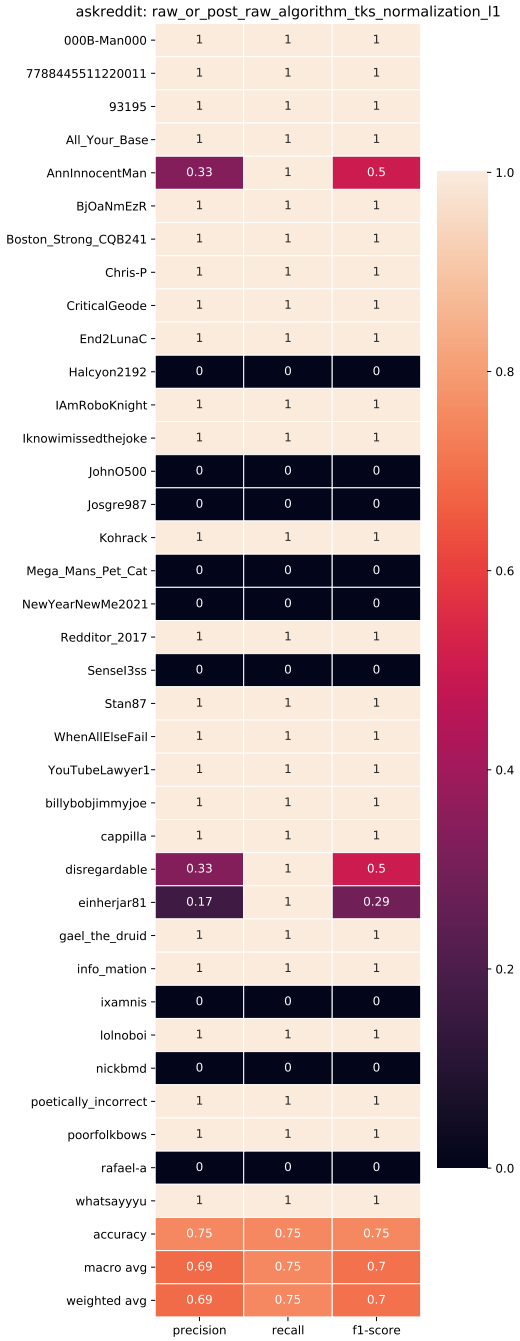

Figure B.13.: Classification report for /r/Askreddit of Method 2.

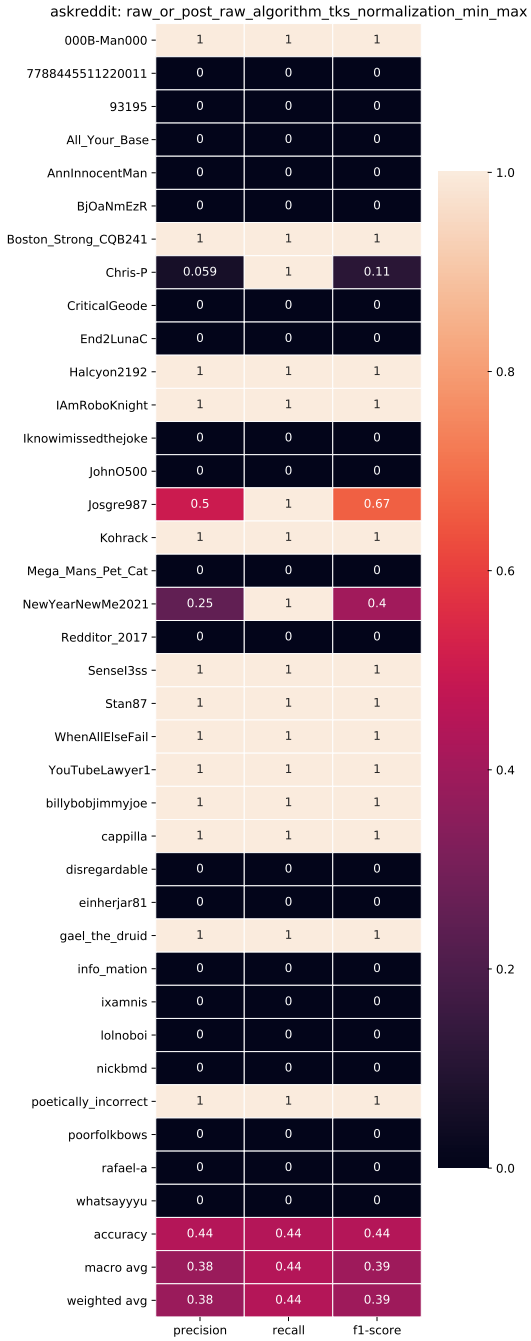

Figure B.14.: Classification report for /r/Askreddit of Method 2.

Bibliography

Abbasi, Ahmed and Hsinchun Chen [2008]. *Writeprints: A Stylometric Approach to Identity-Level Identification and Similarity Detection in Cyberspace*. In: ACM Transactions on Information Systems (TOIS) 26.2 (Apr. 2008), pages 1–29. DOI: 10.1145/1344411.1344413 (cited on page 24).

Afroz, Sadia, Aylin Caliskan Islam, Ariel Stolerman, Rachel Greenstadt, and Damon McCoy [2014]. *Doppelgänger Finder: Taking Stylometry to the Underground*. In: 2014 IEEE Symposium on Security and Privacy. IEEE. May 18, 2014, pages 212–226. DOI: 10.1109/SP.2014.21 (cited on page 5).

Ahonen, Helena, Oskari Heinonen, Mika Klemettinen, and A Inkeri Verkamo [1998]. *Applying Data Mining Techniques for Descriptive Phrase Extraction in Digital Document Collections*. In: Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98. IEEE. Apr. 22, 1998, pages 2–11. DOI: 10.1109/ADL.1998.670374 (cited on pages 3, 11).

Allison, Ben and Louise Guthrie [2008]. *Authorship Attribution of E-mail: Comparing Classifiers over a New Corpus for Evaluation*. In: LREC. Jan. 2008 (cited on page 49).

Argamon, Shlomo and Shlomo Levitan [2005]. *Measuring the Usefulness of Function Words for Authorship Attribution*. In: Proceedings of the 2005 ACH/ALLC Conference. Jan. 2005, pages 4–7 (cited on pages 3, 18, 65).

Argamon, Shlomo, Casey Whitelaw, Paul Chase, Sobhan Raj Hota, Navendu Garg, and Shlomo Levitan [2007]. *Stylistic Text Classification using Functional Lexical Features*. In: Journal of the American Society for Information Science and Technology 58.6 (Feb. 26, 2007), pages 802–822. DOI: 10.1002/asi.20553 (cited on page 25).

Ayres, Jay, Jason Flannick, Johannes Gehrke, and Tomi Yiu [2002]. *Sequential Pattern Mining Using a Bitmap Representation*. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. July 2002, pages 429–435. DOI: 10.1145/775047.775109 (cited on page 22).

Baayen, Harald, Hans Van Halteren, and Fiona Tweedie [1996]. *Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution*. In: Literary and Linguistic Computing 11.3 (Sept. 1996), pages 121–132. DOI: 10.1093/llc/11.3.121 (cited on page 25).

Baumgartner, Jason, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn [2020]. *The Pushshift Reddit Dataset*. In: arXiv preprint arXiv:2001.08435 (2020) (cited on page 37).

Burrows, John [2002]. *'Delta': A Measure of Stylistic Difference and a Guide to Likely Authorship*. In: Literary and Linguistic Computing 17.3 (Sept. 1, 2002), pages 267–287. DOI: 10.1093/llc/17.3.267 (cited on page 17).

Clark, Jonathan H and Charles J Hannon [2007]. *A Classifier System for Author Recognition Using Synonym-Based Features*. In: Mexican International Conference on Artificial Intelligence. Springer. 2007, pages 839–849. DOI: 10.1007/978-3-540-76631-5_80. URL: http://www.cs.cmu.edu/afs/cs/Web/People/jhclark/pubs/MICAI07.pdf (cited on pages 6, 22, 35).

Coyotl-Morales, Rosa María, Luis Villaseñor-Pineda, Manuel Montes-y-Gómez, and Paolo Rosso [2006]. *Authorship Attribution Using Word Sequences*. In: Iberoamerican Congress on Pattern Recognition. Springer. 2006, pages 844–853. DOI: 10.1007/11892755_87 (cited on page 25).

De Vel, Olivier, Alison Anderson, Malcolm Corney, and George Mohay [2001]. *Mining E-mail Content for Author Identification Forensics*. In: ACM Sigmod Record 30.4 (2001), pages 55–64. DOI: 10.1145/604264.604272 (cited on page 25).

Fournier-Viger, Philippe, Antonio Gomariz, Manuel Campos, and Rincy Thomas [2014]. *Fast Vertical Mining of Sequential Patterns Using Co-Occurrence Information*. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer. 2014, pages 40–52. DOI: 10.1007/978-3-319-06608-0_4. URL: http://www.philippe-fournier-viger.com/spmf/PAKDD2014_sequential_pattern_mining_CM-SPADE_CM-SPAM.pdf (cited on pages 22–23).

Fournier-Viger, Philippe, Antonio Gomariz, Ted Gueniche, Espérance Mwamikazi, and Rincy Thomas [2013]. *TKS: Efficient Mining of Top-K Sequential Patterns*. In: International Conference on Advanced Data Mining and Applications. Springer. 2013, pages 109–120. DOI: 10.1007/978-3-642-53914-5_10 (cited on pages 24, 41).

Fournier-Viger, Philippe, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam [2016]. *The SPMF Open-Source Data Mining Library Version 2*. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer. 2016, pages 36–40. DOI: 10.1007/978-3-319-46131-1_8. URL: http://www.philippe-fournier-viger.com/2016_PKDD_SPMF_VERSION2.pdf (cited on pages 12, 41).

Fournier-Viger, Philippe, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas [2017]. *A Survey of Sequential Pattern Mining*. In: Data Science and Pattern Recognition 1.1 (2017), pages 54–77 (cited on pages 13, 22, 24).

Fournier-Viger, Philippe, Cheng-Wei Wu, Antonio Gomariz, and Vincent S Tseng [2014]. *VMSP: Efficient Vertical Mining of Maximal Sequential Patterns*. In: Canadian Conference on Artificial Intelligence. Springer. 2014, pages 83–94. DOI: 10.1007/978-3-319-06483-3_8. URL: http://www.philippe-fournier-viger.com/spmf/VMSP_maximal_sequential_patterns_2014.pdf (cited on page 23).

Frantzeskou, Georgia, Efstathios Stamatatos, Stefanos Gritzalis, Carole E Chaski, and Blake Stephen Howald [2007]. *Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method*. In: International Journal of Digital Evidence 6.1 (Jan. 2007), pages 1–18 (cited on page 49).

Fumarola, Fabio, Pasqua Fabiana Lanotte, Michelangelo Ceci, and Donato Malerba [2015]. *CloFAST: Closed Sequential Pattern Mining Using Sparse and Vertical ID-Lists*. In: Knowledge and Information Systems 48.2 (Oct. 20, 2015), pages 429–463. DOI: 10.1007/s10115-015-0884-x (cited on page 23).

Gomariz, Antonio, Manuel Campos, Roque Marin, and Bart Goethals [2013]. *ClaSP: An Efficient Algorithm for Mining Frequent Closed Sequences*. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer. 2013, pages 50–61. DOI: 10.1007/978-3-642-37453-1_5 (cited on page 23).

Grieve, Jack [2007]. *Quantitative Authorship Attribution: An Evaluation of Techniques.* In: Literary and Linguistic Computing 22.3 (July 26, 2007), pages 251–270. DOI: 10.1093/llc/fqm020 (cited on page 25).

Hirst, Graeme and Ol'ga Feiguina [2007]. *Bigrams of Syntactic Labels for Authorship Discrimination of Short Texts.* In: Literary and Linguistic Computing 22.4 (Oct. 1, 2007), pages 405–417. DOI: 10.1093/llc/fqm023 (cited on page 5).

Hunter, John D [2007]. *Matplotlib: A 2D Graphics Environment.* In: Computing in Science & Engineering 9.3 (2007), pages 90–95 (cited on page 45).

Karlgren, Jussi and Gunnar Eriksson [2007]. *Authors, Genre, and Linguistic Convention.* In: Proceedings from the SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection. 2007 (cited on page 25).

Kern, Roman [2013]. *Grammar Checker Features for Author Identification and Author Profiling.* In: CLEF 2013 Evaluation Labs and Workshop–Working Notes Papers. Citeseer, 2013. DOI: 10.1.1.666.9989 (cited on page 25).

Kešelj, Vlado, Fuchun Peng, Nick Cercone, and Calvin Thomas [2003]. *N-Gram-Based Author Profiles for Authorship Attribution.* In: Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING. Volume 3. sn. 2003, pages 255–264. URL: https://web.cs.dal.ca/~vlado/papers/pacling03.pdf (cited on pages 9, 17, 25, 64).

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlayand, et al. [2016]. *Jupyter Notebooks - A Publishing Format for Reproducible Computational Workflows.* In: ELPUB. 2016, pages 87–90. DOI: 10.3233/978-1-61499-649-1-87 (cited on page 37).

Koppel, Moshe and Jonathan Schler [2003]. *Exploiting Stylistic Idiosyncrasies for Authorship Attribution.* In: Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis. Volume 69. 2003, pages 72–80 (cited on page 25).

Koppel, Moshe, Jonathan Schler, and Shlomo Argamon [2011]. *Authorship Attribution in the Wild.* In: Language Resources and Evaluation 45.1 (Mar. 2011), pages 83–94. DOI: 10.1007/s10579-009-9111-2 (cited on page 49).

Li, Chun and Jianyong Wang [2008]. *Efficiently Mining Closed Subsequences with Gap Constraints*. In: Proceedings of the 2008 SIAM International Conference on Data Mining. SIAM. 2008, pages 313–322. DOI: 10.1137/1.9781611972788.28 (cited on page 41).

Li, Jiexun, Rong Zheng, and Hsinchun Chen [2006]. *From Fingerprint to Writeprint*. In: Communications of the ACM 49.4 (Apr. 2006), pages 76–82. DOI: 10.1145/1121949.1121951 (cited on page 25).

Loper, Edward and Steven Bird [2002]. *NLTK: the Natural Language Toolkit*. In: CoRR cs.CL/0205028 (June 2002). DOI: 10.3115/1118108.1118117 (cited on pages 11, 33).

Marton, Yuval, Ning Wu, and Lisa Hellerstein [2005]. *On Compression-Based Text Classification*. In: European Conference on Information Retrieval. Springer. 2005, pages 300–314. DOI: 10.1007/978-3-540-31865-1_22 (cited on page 17).

Marujo, Luìs, Anatole Gershman, Jaime Carbonell, Robert Frederking, and João P Neto [2013]. *Supervised Topical Key Phrase Extraction of News Stories Using Crowdsourcing, Light Filtering and Co-reference Normalization*. In: arXiv preprint arXiv:1306.4886 (June 20, 2013) (cited on pages 2–3, 11).

Muhr, Markus, Roman Kern, Mario Zechner, and Michael Granitzer [2010]. *External and Intrinsic Plagiarism Detection Using a Cross-Lingual Retrieval and Segmentation System*. In: Notebook Papers of CLEF 2010 LABs and Workshops. Oct. 2010 (cited on page 22).

Neal, Tempestt, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard [2017]. *Surveying Stylometry Techniques and Applications*. In: ACM Comput. Surv. 50.6 (Nov. 2017). ISSN: 0360-0300. DOI: 10.1145/3132039 (cited on page 79).

Overdorf, Rebekah and Rachel Greenstadt [2016]. *Blogs, Twitter Feeds, and Reddit Comments: Cross-Domain Authorship Attribution*. In: Proceedings on Privacy Enhancing Technologies 2016.3 (May 6, 2016), pages 155–171. DOI: 10.1515/popets-2016-0021 (cited on pages 5, 21).

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg [2011]. *Scikit-Learn: Machine Learning in Python*. In: The

Journal of Machine Learning Research 12 (Oct. 11, 2011), pages 2825–2830 (cited on pages 12, 40).

Pei, Jian, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu [2004]. *Mining Sequential Patterns by Pattern-Growth: The Prefixspan Approach*. In: IEEE Transactions on Knowledge and Data Engineering 16.11 (Oct. 4, 2004), pages 1424–1440. DOI: 10.1109/TKDE.2004.77 (cited on page 22).

Peng, Fuchun, Dale Schuurmans, Shaojun Wang, and Vlado Kešelj [2003]. *Language Independent Authorship Attribution Using Character Level Language Models*. In: Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics-Volume 1. Association for Computational Linguistics. Apr. 2003, pages 267–274. DOI: 10.3115/1067807.1067843 (cited on page 25).

Petitjean, François, Tao Li, Nikolaj Tatti, and Geoffrey I Webb [2016]. *Skopus: Mining Top-K Sequential Patterns under Leverage*. In: Data Mining and Knowledge Discovery 30.5 (June 14, 2016), pages 1086–1111. DOI: 10.1007/s10618-016-0467-9 (cited on page 24).

Ramos, Juan [2003]. *Using TF-IDF to Determine Word Relevance in Document Queries*. In: Proceedings of the First Instructional Conference on Machine Learning. Volume 242. Piscataway, NJ. Jan. 2003, pages 133–142 (cited on page 13).

Rexha, Andi, Mark Kröll, Hermann Ziak, and Roman Kern [2018]. *Authorship Identification of Documents with High Content Similarity*. In: Scientometrics 115.1 (Feb. 2, 2018), pages 223–237. DOI: 10.1007/s11192-018-2661-6 (cited on page 2).

Riloff, Ellen and Wendy Lehnert [1994]. *Information Extraction as a Basis for High-Precision Text Classification*. In: ACM Transactions on Information Systems (TOIS) 12.3 (July 1994), pages 296–333. DOI: 10.1145/183422.183428 (cited on page 11).

Ruder, Sebastian, Parsa Ghaffari, and John G Breslin [2016]. *Character-Level and Multi-Channel Convolutional Neural Networks for Large-Scale Authorship Attribution*. In: arXiv preprint arXiv:1609.06686 (Sept. 21, 2016) (cited on pages 5, 20, 48–49, 71).

Sanderson, Conrad and Simon Guenter [2006]. *Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: An Investigation.* In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. July 2006, pages 482–491 (cited on pages 17, 25).

Sebastiani, Fabrizio [2002]. *Machine Learning in Automated Text Categorization.* In: ACM Computing Surveys (CSUR) 34.1 (2002), pages 1–47. DOI: 10.1145/505282.505283 (cited on pages 17, 25).

Seroussi, Yanir, Ingrid Zukerman, and Fabian Bohnert [2011]. *Authorship Attribution with Latent Dirichlet Allocation.* In: Proceedings of the Fifteenth Conference on Computational Natural Language Learning. June 2011, pages 181–189 (cited on page 49).

Srikant, Ramakrishnan and Rakesh Agrawal [1996]. *Mining Sequential Patterns: Generalizations and Performance Improvements.* In: International Conference on Extending Database Technology. Springer. 1996, pages 1–17. DOI: 10.1007/BFb0014140 (cited on page 22).

Stamatatos, Efstathios [2009]. *A Survey of Modern Authorship Attribution Methods.* In: Journal of the American Society for Information Science and Technology 60.3 (2009), pages 538–556. DOI: 10.1002/asi.21001 (cited on pages 3, 11, 16, 24).

Stamatatos, Efstathios, Nikos Fakotakis, and Georgios Kokkinakis [2000]. *Automatic Text Categorization in Terms of Genre and Author.* In: Computational Linguistics 26.4 (Dec. 2000), pages 471–495. DOI: 10.1162/089120100750105920 (cited on page 25).

Stamatatos, Efstathios, Nikos Fakotakis, and Georgios Kokkinakis [2001]. *Computer-Based Authorship Attribution Without Lexical Measures.* In: Computers and the Humanities 35.2 (May 2001), pages 193–214. DOI: 10.1023/A:1002681919510 (cited on page 25).

Stolerman, Ariel, Rebekah Overdorf, Sadia Afroz, and Rachel Greenstadt [2013]. *Classify, But Verify: Breaking the Closed-World Assumption in Stylometric Authorship Attribution.* In: IFIP Working Group. Volume 11. 2013, page 64. URL: http://www.stolerman.net/papers/classify_verify_ifip-wg11.9-2014.pdf (cited on page 18).

Suman, Chanchal, Sriparna Saha, Pushpak Bhattacharyya, and Rohit Shyamkant Chaudhari [2020]. *Emoji Helps! A Multi-Modal Siamese Architecture for Tweet User Verification*. In: Cognitive Computation (Mar. 2, 2020), pages 1–16. DOI: 10.1007/s12559-020-09715-7 (cited on pages 6, 21).

Van Halteren, Hans [2007]. *Author Verification by Linguistic Profiling: An Exploration of the Parameter Space*. In: ACM Transactions on Speech and Language Processing (TSLP) 4.1 (Feb. 2007), pages 1–17. DOI: 10.1145/1187415.1187416 (cited on page 16).

Wang, Jianyong and Jiawei Han [2004]. *BIDE: Efficient Mining of Frequent Closed Sequences*. In: Proceedings. 20th International Conference on Data Engineering. IEEE. Apr. 2, 2004, pages 79–90. DOI: 10.1109/ICDE.2004.1319986 (cited on pages 23, 41).

Witten, Ian H [2004]. *Text Mining*. 2004. URL: www.cs.waikato.ac.nz/~ihw/papers/04-IHW-Textmining.pdf (cited on page 11).

Yan, Xifeng, Jiawei Han, and Ramin Afshar [2003]. *CloSpan: Mining: Closed Sequential Patterns in Large Datasets*. In: Proceedings of the 2003 SIAM International Conference on Data Mining. SIAM. 2003, pages 166–177. DOI: 10.1137/1.9781611972733.15 (cited on page 23).

Zaki, Mohammed J [2001]. *SPADE: An Efficient Algorithm for Mining Frequent Sequences*. In: Machine Learning 42.1-2 (Jan. 2001), pages 31–60. DOI: 10.1023/A:1007652502315 (cited on page 22).

Zhang, Yongzheng, Nur Zincir-Heywood, and Evangelos Milios [2005]. *Narrative Text Classification for Automatic Key Phrase Extraction in Web Document Corpora*. In: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management (WIDM). Nov. 5, 2005, pages 51–58 (cited on page 11).

Zhao, Ying and Justin Zobel [2005]. *Effective and Scalable Authorship Attribution Using Function Words*. In: Asia Information Retrieval Symposium. Springer. 2005, pages 174–189. DOI: 10.1007/11562382_14 (cited on page 17).

Zhao, Ying and Justin Zobel [2007]. *Searching with Style: Authorship Attribution in Classic Literature*. In: Proceedings of the Thirtieth Australasian Conference on

Computer Science - Volume 62. Australian Computer Society, Inc. Jan. 2007, pages 59–68. DOI: 10.5555/1273749.1273757 (cited on page 25).

Zheng, Rong, Jiexun Li, Hsinchun Chen, and Zan Huang [2006]. *A Framework for Authorship Identification of Online Messages: Writing-Style Features and Classification Techniques*. In: Journal of the American Society for Information Science and Technology 57.3 (Feb. 1, 2006), pages 378–393. DOI: 10.1002/asi.20316 (cited on page 25).