Finding Argument Fragments on Social Media with Corpus Queries and LLMs

Nathan Dykes[1](✉), Stephanie Evert[1], Philipp Heinrich[1], Merlin Humml[2], and Lutz Schröder[2]

[1] Chair of Computational Corpus Linguistics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Bismarckstr. 6, 91054 Erlangen, Germany
{nathan.dykes,stephanie.evert,philipp.heinrich}@fau.de

[2] Chair of Theoretical Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
{merlin.humml,lutz.schroder}@fau.de

Abstract. We are concerned with extracting argumentative fragments from social media, exemplified with a case study on a large corpus of English tweets about the UK Brexit referendum in 2016. Our overall approach is to parse the corpus using dedicated corpus queries that fill designated slots in predefined logical patterns. We present an inventory of logical patterns and corresponding queries, which have been carefully designed and refined. While a gold standard of substantial size is difficult to obtain by manual annotation, our queries can retrieve hundreds of thousands of examples with high precision. We show how queries can be combined to extract complex nested statements relevant to argumentation. We also show how to proceed for applications needing higher recall: high-precision query matches can be used as training data for an LLM classifier, and the trade-off between precision and recall can be freely adjusted with its cutoff threshold.

Keywords: Argument extraction · Semantic parsing · Corpus queries · Social media · LLMs

1 Introduction

We report on a methodology for extracting arguments from posts in social media, with the overarching aim of gaining an overview of arguments and views being voiced. Argumentation on social media is characterized by a high degree of informality, which makes our endeavour, viz. mapping the argumentative landscape on social media, particularly difficult.

In our central case study, we analyse English tweets about the Brexit referendum in 2016. Tweets are particularly challenging because they are too short to contain fully structured arguments (i.e., premises, conclusions, and links relating the argument parts), see e.g. Bhatti, Ahmad, and Park [3]. Therefore, we aim for a semantically precise method to capture argument fragments by parsing them into a pre-defined but extensible set of logical patterns, i.e. formulae with placeholders in dedicated modal logics. For each logical pattern, we formulate multiple dedicated corpus queries [11] reflecting different linguistic realizations of the same type of statement. It is expected that such an approach will have comparatively low recall but (usually) very high precision. This in fact suits our intention to map general tendencies in the argumentative landscape and reconstruct complete arguments from precisely parsed fragments, rather than extracting each individual instance of an argument type.

In this paper, we demonstrate our overall workflow, query development and evaluation results; in particular confirming that we arrive at high precision through corpus queries. In a subsequent extension of the approach, we experiment with *hierarchical* patterns and associated queries, where we look for matches of an inner pattern in the text spans corresponding to placeholders of an outer pattern. Moreover, we present an approach that uses query results as training data for an LLM classifier in order to change the precision-recall trade-off, with promising results.

Related work Work on argument mining in social media often focuses on graphical structure (e.g. [1,15]; see also Lytos et al. [19] for a survey, and Lytos et al. [20] for an example of a recent, purely data-driven, approach), and has highlighted linguistic and logical challenges [4,7,14].

Our high-precision approach based on logical patterns and queries appears to be new as such, and is distinct in particular from text mining with knowledge patterns (e.g. [6,23]). Work on the extraction of counterfactuals [32] follows partly similar methods, but uses regular expressions instead of linguistically informed corpus queries.

Recent work on argument mining in Twitter has concentrated on identifying high-level categories such as *argumentative* vs. *non-argumentation*, *factual* vs. *opinion*, *claim* vs. *support* vs. *rebuttal*, etc., which specify the general role of each tweet in an argument (e.g. [2,12,30]). This is much more coarse-grained than (and also fundamentally different from) our approach of extracting the content of argumentative fragments in the form of logical patterns.

NLP approaches to argument mining often focus on automatic classification of such categories by training machine learning algorithms (e.g. a support vector machine or logistic regression), see e.g. [5] for a survey. However, with only a handful of positive examples every one hundred tweets (see Sect. 3.2), obtaining a sufficient amount of training examples is prohibitively expensive. Recent work on large language models (LLMs), on the other hand, promises to leverage linguistic knowledge derived from unlabeled data; these models only have to be *fine-tuned* on the classification task at hand [see e.g. 25,26]. We will show in this paper that fine-tuning a general-purpose LLM on a handful of positive examples does not yield satisfying results. A more competitive approach are frameworks for few-shot learning. In SetFit [33], a pre-trained Sentence Transformers [27] model is first fine-tuned on a number of contrastive pairs of labelled texts and then used to encode the training data. Finally, a text classification head is trained using the encoded data.[1]

Our combined approach outlined below, i.e. leveraging corpus queries for training an LLM, is in fact similar to *data augmentation* [13], i.e. increasing the diversity of training examples without collecting new data. However, data augmentation usually works by creating artifical examples that are very similar to the original positive examples (or are made up altogether), while our approach uses only authentic examples as training data. Finally, in the case that high-quality training data are not available, one could use "weak labeled data" [31], applying coarse heuristics to extract training examples while allowing for a significant amount of noise. This also bears similarities to our approach, but is fundamentally different from our high-precision, low-recall strategy.

2 Argumentative Fragments

Given the complexity of natural language argumentation, we approach argument mining in a piece-by-piece manner, aiming to parse argument fragments from our inventory of predefined logical patterns by means of high-precision corpus queries. This means we define logical patterns expressing forms of propositions used in arguments which then have multiple corresponding corpus queries each covering multiple syntactic realizations to express such a proposition.

2.1 An Inventory of Logical Patterns

Starting from an analysis of argument schemes in the style of Reed, and Macagno [34], we created an initial inventory of logical patterns for argumentation. This inventory was extended with additional patterns that were common in our data but outside of the standard catalogue of argumentation schemes, to adapt to the informality of arguments on social media.

The patterns are formulae with sorted placeholders in dedicated modal logics. A typical example is the *desire* pattern $D_{\{?0:entity\}}\{?1 : formula\}$ expressing that entity ?0 wants formula ?1 to become true. Note that the sort *entity* does not require the expression to evaluate to a single entity but instead describes an abstract group of entities in the sense of Humml and Schröder [17]. Patterns can also go beyond single modality statements to more complicated formulae or even sets of formulae (or equivalently conjunctions) like, e.g., the *group knowledge* pattern $K_{\{?0:entity\}}(\{?1 : formula\}); (?2) \implies (?0)$ expressing that entity ?2 is part of entity ?0 whose members know that formula ?1 is true. The underlying semantics of abstract group knowledge then implies that ?2 also knows ?1, i.e. $K_{\{?2:entity\}}(?1)$. This pattern could, e.g., be an indicator for an argument from *Position to Know* [34]. The logical framework we use has been described in more detail in earlier work [8,9]. As indicated above, the overall character of the

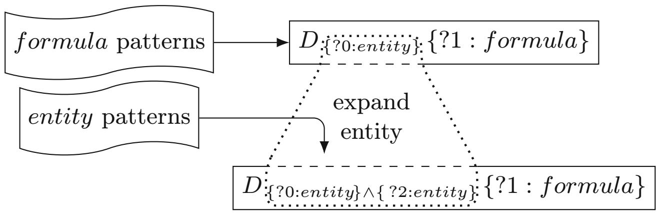
[1] Other state-of-the-art methods such as T-Few [18] might yield even better results, but SetFit is a convenient and widely-used framework that does not require any prompt-engineering.

representation logic is *modal* in the sense that it features operators expressing that statements hold in a certain way; e.g. the operators D and K ('desires' / 'knows') featuring in the above examples are modal operators.

Our motivation for extracting argumentative content in the form of logical statements is to leverage automated reasoners to aid in reconstructing complete arguments. In everyday argumentation, it is uncommon and even socially unacceptable to give detailed arguments that mention every reasoning step and every premise. Instead, dialogical argumentation relies on shared common knowledge between the dialogue participants to complete the missing parts of arguments. In our processing pipeline, the logical reasoner and a knowledge base are eventually intended to take the place of the human reasoner trying to fill in the missing pieces of the arguments. For example the reasoner might combine desire statements into larger desired states of the world. In our corpus of tweets the queries retrieve two desire statements attributed to Cameron: "PROOF Cameron WANTS Turkey to join the EU [...]" "Ersatz 'reform deal' proves Cameron always wanted the UK to stay in the EU [...]" The reasoner would then draw the conclusion that Cameron wants both the UK and Turkey to be in the EU.

2.2 Nested Patterns

Combining patterns from our inventory can yield many variants of more complex statements. We follow a recursive approach to extracting relevant information from selected pattern combinations. Empirically, we apply corpus queries to the text spans matching placeholders of an "outer" pattern in order to find matches of further "inner" patterns. The sorting discipline on placeholders then implies a corresponding sorting discipline on the patterns themselves with different logic syntaxes used in patterns of different sorts. For example, a formula describing an entity will employ different modalities than a formula defining an action or a truth statement. In Fig. 1, the entity slot in the *desire* pattern is expanded by filling in a more complicated entity expression from the set of entity patterns; in the example, this expression denotes the intersection of two entities (remember that entities are abstract groups). Similarly, the placeholder ?1, which represents a truth statement, could be expanded with a more concrete pattern, such as *membership* [8]. A concrete realization of the doubly extended pattern would be an expression like "trustworthy economists favour the UK being in the EU", which



formula patterns
entity patterns
$D_{\{?0:entity\}}\{?1 : formula\}$
expand entity
$D_{\{?0:entity\}\wedge\{?2:entity\}}\{?1 : formula\}$

Fig. 1. A nested pattern

would be expressed as the formula $D_{\mathsf{isTrustworthy} \wedge \mathsf{isEconomist}}(\mathsf{UK} \rightarrow \mathsf{EU})$. In practice, this approach is implemented by hierarchical queries as discussed in Sect. 5.

3 Data

3.1 Corpus and Linguistic Annotation

We use the data set collected by Dykes et al. [9], which consists of tweets containing the token *brexit* (case-insensitive, with or without hashtag marker) collected before and after the UK Brexit referendum in 2016. Our final corpus only includes original tweets and has been filtered with a deduplication algorithm (which disregards case differences, @-mentions, URLs, and hashtags). It amounts to ca. 4.3 million tweets with a total size of ca. 80 million tokens, most of them posted close to the actual referendum on June 23, 2016.

The linguistic annotation pipeline follows [8], comprising Ark TweetNLP [22][2] for simple PoS tags, the OSU Twitter NLP tools [28][3] for Penn-style PoS tags and named entity recognition, and a lemmatizer based on Minnen, Carroll, and Pearce [21]. Sentence boundaries are inserted using SoMaJo [24][4]. Note that these systems generate different tokenization layers, which are reconciled in post-processing. The final corpus is indexed using the IMS Open Corpus Workbench [10].[5]

3.2 Manual Annotation of Argument Fragments

Several student assistants were hired to annotate relevant argument fragments in two random corpus samples: `pre` contains 785 tweets from before the referendum, `post` contains 973 tweets from August 21, 2016 (i.e. after the referendum). Doubtful cases were discussed with the project members, and all disagreements in annotation were adjudicated on a regular basis. Note that prevalence of patterns is low: only about 5–7% of all tweets contain *desire* and 7–10% contain *membership*, cf. Table 2. Additionally, random samples of query matches (data sets `matches`) were annotated, amounting to a total of 3997 tweets for the *desire* pattern and 1005 tweets for *membership*.

Annotation was highly time-consuming, thus it was unrealistic to obtain a sufficient number of examples to train a machine-learning classifier to detect patterns automatically. Several factors contributed to this challenge: Firstly, the aforementioned low prevalence of patterns meant that annotators needed to check vast numbers of tweets. Secondly, the linguistic realizations that do occur take many different forms even within the same pattern, which made it easy to miss relevant cases. Thirdly, despite working with detailed annotation guidelines, the decision of whether a given expression fits a particular logical formula or not

[2] http://www.cs.cmu.edu/~ark/TweetNLP/.
[3] https://github.com/aritter/twitter_nlp.
[4] https://github.com/tsproisl/SoMaJo.
[5] https://cwb.sourceforge.io/.

still proved difficult. These difficulties are reflected in the (sometimes low) inter-annotator agreement scores in Table 1. It is worth noting that the annotators where instructed to err on the side of annotating doubtful cases positively, with corner cases included in the subsequent adjudication process.

Table 1. Kappa scores for the three most annotated patterns. *E*, *M*, and *V* represent three independent student assistants, *gold* was obtained in a subsequent adjudication process.

(a) *desire*

	E	M	V
M	0.39	-	-
V	0.79	0.39	-
gold	0.46	0.86	0.45

(b) *membership*

	E	M	V
M	0.66	-	-
V	0.59	0.62	-
gold	0.80	0.72	0.63

(c) *opposition*

	E	M	V
M	0.35	-	-
V	0.59	0.42	-
gold	0.36	0.76	0.46

4 Corpus Queries

4.1 Methods

The manually annotated instances of our argument patterns serve as templates for specialized corpus queries. For the case of *desire*, recall that our aim is to find realisations of the formula $D_{\{?0:entity\}}\{?1 : formula\}$. The following statements are examples of posts expressing a *desire* according to our guidelines:

1. "without giving u reasons for u to argue with, I think *I'm in favour of an exit*!!"
2. "*Several key @vote_leave folks on record wanting to privatise #NHS* & #Brexit #Tory ministers never showed any concern for NHS @stariep"
3. "@SadiqKhan Sir, are *you in favor of #Brexit*?"
4. "eAndrew Neil is chair of *@spectator which has come out for #Brexit* How can @afneil still be allowed control of #BBCSDP #BBCDP?"
5. "*Bryan Adams is in favour of Brexit.*"

While these examples all correspond to our formula, they are clearly not identical linguistically. Variation occurs in terms of what the entity and the formula refer to, as well as how each concept is expressed regarding lexis and syntax. Nevertheless, examples 1, 3 and 5 can be generalised on the linguistic level to *ENTITY is in favour of FORMULA*. Based on such similarities, we constructed the following query to extract further similar instances of *desire* from the overall corpus:

```
@0[::] /entity_np_actor[] @1[::]
[xpos=‘‘MD’’ | lemma=‘‘be|have’’ | upos=‘‘ADJ|ADV’’]*
[lemma=‘‘in’’] [upos=‘‘ADJ’’]* [lemma=$nouns_desire]
[lemma=‘‘for|of|pro|that|to’’]
@2[::] (/entity_np_all[] | [xpos=‘‘VBG’’]) (/lexical_words[])
    ↪ * @3[::]
```

Our queries are written in the query language [11] of the corpus query processor (CQP) of the IMS Open Corpus Workbench (CWB, [10]), enabling efficient execution of complex queries in large corpora. The query language is based on regular expressions over token descriptions, which are Boolean expressions of attribute-value pairs (where values can again be matched by regular expressions). For example, `[lemma=‘‘be’’]` retrieves all forms of *BE* (*be, am, are, is, was, were, been, being*), and `[upos=‘‘ADJ’’]*` retrieves sequences of adjectives. Additionally, structural annotation elements (such as tweets, paragraphs or sentences) can be matched by XML tags, e.g. `<tweet> []* </tweet>` for a complete tweet.

The most important parts of this query are the slot fillers. For *desire*, they represent the ENTITY and FORMULA slots. In the corresponding query, the tokens belonging to a given slot are enclosed in target markers: `@0[::] ... ↪ @1[::]` (ENTITY) and `@2[::] ... @3[::]` (FORMULA). The ENTITY is modelled with a CQP macro `/entity_np_actor[]`, which expands to match noun phrases containing personal pronouns, proper names, or nouns from a word list referencing people or organizations (e.g. *politician* or *party*). Limiting the noun phrase in this way ensures that the expression in the ENTITY slot can reasonably be expected to express a meaningful desire. While we will necessarily lose some recall with this restriction, a more flexible ENTITY slot filler would compromise precision too much. However, the word lists were extended using word embeddings, which we used to suggest distributionally similar items to the ones that had been collected manually. The macro `/entity_np_all[]` in the FORMULA region matches a much more general noun phrase, since FORMULA can refer to a wider range of concepts. Alternatively, this slot can be realized with a verb in gerund form (e.g. *to be in favour of exiting*), followed by an arbitrary number of content words within the same tweet. The middle part of the query provides linguistic structure to ensure that it actually matches an expression of *desire*. After optional modifiers, its main part is *in favour/hope/support/... for/of/....*

Development Environment. There are two major shortcomings to the main CWB user interface CQPweb [16], which we initially used for query development (similar limitations apply to other tools like AntConcc[6] or Sketch Engine[7]). Firstly, when writing a large repertoire of queries, reusable elements like word lists and macros need to be managed efficiently. While CWB can easily read

[6] https://www.laurenceanthony.net/software.
[7] https://www.sketchengine.eu/.

macros and word lists from plain text files, CQPweb does not provide access to these files.

More importantly, these tools are designed with traditional corpus studies in mind, which typically use much shorter queries. Accordingly, functionality for displaying and sorting query results is usually optimised for single words and short expressions. In our usage scenario, i.e. argument queries, it is crucial to mark and highlight multiple positions within query matches. Queries that retrieve surface realizations of the *desire* pattern need to specify two *slots* (text spans) representing the ENTITY and FORMULA placeholders, respectively.

It has only recently become possible in version 3.4.16 of CWB to mark more than a single position inside a query match (in addition to start and end of the match), using anchors `@0`, ..., `@9`. This new feature requires support from a wrapper application, though, which has to run every query up to 5 times, collecting two anchor positions in each step. We provide the Python library *cwb-ccc*[8], which includes such a wrapper.[9] Anchor positions can also be adjusted by an integer offset; this is especially helpful if the query contains optional elements (with quantifiers `?`, `+` or `*`).

Since developing corpus queries is an iterative hermeneutic process, carefully balancing precision and recall for the task at hand, it would be very inconvenient to run a wrapper from the command line and collect its results whenever a query is modified. We thus developed *Spheroscope*[10], a web app specifically dedicated to the iterative development of corpus queries.

Here, queries can use an arbitrary number of word lists and macros, which can be stored and re-used via the user interface. The interface also enables users to obtain the frequencies of all words from a word list for any given corpus. Additionally, semantically similar words can be suggested for semi-automatic extension of word lists. For semantic similarity, we use custom word embeddings trained on a larger, independent sample of English tweets. Similar items can be sorted by their corpus frequency or by cosine similarity (by default, up to 200 items are displayed, hapax legomena excluded). Similarly, macros can be defined, named, stored, inspected (frequency breakdown), and reused via the user interface.

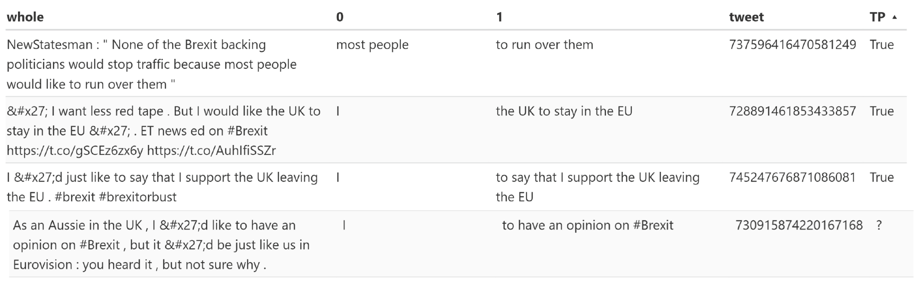
Iterative Query Development. In order to incorporate feedback from manual annotation and to reflect our developing understanding of possible realizations of our continuously refined inventory of argument fragments, query development is necessarily iterative. This affects evaluation, since precision and recall need to be reassessed with every change to the queries. Recall can only be measured on random subsets of tweets; precision can be assessed qualitatively by reading concordance lines of query matches as well as quantitatively by using labelled examples. The development environment thus directly indicates for each query

[8] https://pypi.org/project/cwb-ccc/.

[9] The module provides additional functionality for tradtional corpus linguistic tasks such as keyword and collocation analysis. It is now the official Python API to CWB.

[10] https://github.com/ausgerechnet/spheroscope.

match whether it is a true positive (if the tweet is contained in any gold standard), cf. Figure 2.



whole	0	1	tweet	TP ▲
NewStatesman : " None of the Brexit backing politicians would stop traffic because most people would like to run over them "	most people	to run over them	737596416470581249	True
&#x27; I want less red tape . But I would like the UK to stay in the EU &#x27; . ET news ed on #Brexit https://t.co/gSCEz6zx6y https://t.co/AuhIfiSSZr	I	the UK to stay in the EU	728891461853433857	True
I &#x27;d just like to say that I support the UK leaving the EU . #brexit #brexitorbust	I	to say that I support the UK leaving the EU	745247676871086081	True
As an Aussie in the UK , I &#x27;d like to have an opinion on #Brexit , but it &#x27;d be just like us in Eurovision : you heard it , but not sure why .	I	to have an opinion on #Brexit	730915874220167168	?

Fig. 2. Concordance View. For each query result, the actual text of the whole tweet is displayed, as well as the surface realizations of each defined slot. Additionally, column TP indicates whether the match is a *true positive* (True), *false positive* (False), or unknown (?) as the tweet is not in the gold standard.

4.2 Evaluation and Discussion

Results for our current version of the queries for *desire* and *membership* can be found in Table 2. Note that the most reliable estimates for precision can be taken from the annotation of actual query `matches`, whereas recall is most accurately estimated from `post` (since `pre` was used in the course of developing queries).

Table 2. Pattern-based evaluation of query approach for patterns *desire* and *membership* on different data sets alongside prevalence values. Recall of querying can most reliably be estimated from `post`, while precision can most reliably estimated on actual query matches (indicated in bold).

pattern	data set	prevalence	TN	FN	TP	FP	precision	recall	support
desire	`pre`	0.07	721	31	30	3	0.91	0.49	785
	`post`	0.05	923	25	19	6	0.76	**0.43**	973
	`matches`				2361	97	**0.96**		175022
membership	`pre`	0.10	705	62	13	5	0.72	0.17	785
	`post`	0.07	901	65	6	1	0.86	**0.08**	973
	`matches`				952	53	**0.95**		54412

As noted above, corpus queries are abstractions of the manually identified hits for a given pattern in the gold standard (based on `pre`). While they help us to find several hundred thousands of instances of *desire* on the corpus, their recall is restricted to maximize precision. In this section, we explore the nature of potential recall issues in more detail.

Statistical measures on precision and recall only show part of the picture: since our logical patterns are much more abstract than their realizations in the corpus, it is likely that, for politically relevant statements, even if an individual instance was missed, we may still have found other tweets containing equivalent information on the same entities and concepts. We therefore conducted a qualitative evaluation of tweets from our gold standard that were marked as *desire*, but were not retrieved by any of the queries written for this pattern.

We found a total of 65 false negatives for *desire* in our gold standard. Slightly more than half of these instances were excluded from further analysis because they were either no longer part of the corpus (9), they were assigned to sub-patterns of *desire* (6), or because they were categorized as purely situational, e.g. because the entity was the speaker (22). The remaining 28 tweets were left as genuinely relevant statements to examine in more detail.

Typical reasons why a tweet was missed by the queries include both syntactic and lexical properties. On the syntactic side, we found long distances between the entity and the formula (**Banks** now call for 'passporting'; to be ditched and instead want a **'hard Brexit'**). Similar issues relate to tweets containing uncommon vocabulary or typos in slots using word lists (**Britian** want Brexit to go away).

In order to see whether tweets with equivalent content were present in the query results, we searched for the ENTITY for each of these false negatives within the query matches for *desire* and read the matches. For 23 out of 28 cases, an exact or very close match was found. For instance, while *Dennis Skinner for Brexit !!! YASSSSSSS !!!!* was missed, our *desire* queries matched *Dennis Skinner backs Brexit for democracy* and *Labour MPs Dennis Skinner and John Mann back Brexit.* Occasionally, the ENTITY was expressed in slightly different ways, but could still be related back to the same referent with contextual understanding. This incudes the following tweet, which we missed due to its relative clause: *Andrew Neil is chair of @spectator which has come out for #Brexit.* While our query results for *desire* did not include the @spectator account as the ENTITY, several instances referred to *The Spectator.*

For five relevant false negatives, we could not find a very similar equivalent in the query results. In some cases, the ENTITY was a relevant actor, but still infrequent in the corpus (*Globalists R desperate to abolish nations & families*). Alternatively, the FORMULA was too vague to be reconstructed (*You can sense people revelling in it on some level. Desperate for* ***something to come out that proves Farage or Leave or Brexit did this***).

In summary, this evaluation suggests that, at least for statements that have been expressed by a reasonably large number of users, the queries mostly still find logically equivalent propositions even where individual realisations are missed due to unusual wording or syntax.

5 Hierarchical Queries

5.1 Methods

Besides running on the overall corpus, queries can be nested to find argument fragments *within* the slots of larger fragments. For instance, if we run queries for the *membership* pattern on the FORMULA slot of our *desire* queries, we expect the results to be a hierarchical combination of the two patterns (e.g. *I want Britain to be in the EU*). The technical implementation consists of four steps:

1. Run queries for a given pattern and extract the text spans matching the slots of individual placeholders.
2. Form one sub-corpus per placeholder slot containing only these text spans.
3. Run queries for patterns of the correct sort on each sub-corpus.
4. Further instantiate the extracted formulae by substituting placeholders in the partial formula of the outer query with the formula obtained from the sub-query.

It is obvious that, in order for the results to be meaningfully interpretable, the inner query needs to be contained within the relevant slot of the outer query. However, it is less clear whether one should only consider exact matches (where the *membership* statement matches the entire FORMULA slot of *desire*) or also accept cases where only a part of the outer slot is matched by the inner query.

5.2 Evaluation

Therefore, we evaluated hierarchical queries for the combined pattern *ENTITY desires MEMBERSHIP*.

Table 3. Evaluation of hierarchical queries for *ENTITY desires MEMBERSHIP* for different positions of the inner query in the outer slot

inner/outer	#matches	TP	FP	correct example
exact	446	48	2	Donald Trump Supports ***The UK Leaving The European Union***
left	260	33	17	he would support ***Texas leaving the US*** *and becoming an independent state*
right	215	11	39	let's hope *we get a strong turnout on the day and* ***we leave the EU***
within	163	0	50	—

Table 3 shows the number of matches for each position of the inner query in the slot of the outer query, as well as the number of true and false positives in a manual validation of a random sample of 50 tweets for each position. *Exact* matches are almost always correct instances of the combined pattern. The majority of cases where the inner query match is only aligned with the *left* slot boundary are also correct, although the precision is considerably lower than for complete matches.

5.3 Discussion

A common type of false positive in this set of tweets is due to the ambiguity of the word *join*, e.g. *We wish the Netherlands will join us soon with a #Nexit and kick out their anti-democratic rulers.* Our combined query misinterprets this tweet as the Netherlands becoming a member of *us*. Most matches where the end of the inner query result is aligned with the *right* slot boundary are mostly false positives. Even though such cases usually do involve statements about membership, the membership assertion is typically embedded in some other statement that would also need to be parsed for a meaningful interpretation (e.g. *I really hope the Brits understand how turbulent Europe will be if* ***UK leaves EU***). Finally, none of the cases where the inner query match is strictly contained *within* the outer query's slot were true positives. Similarly to the *right* overlap, while the formula was typically related to membership, a multi-step parsing process involving additional patterns would have been required (*I wished I knew if* ***UK leaving the EU*** *is good or bad*).

6 Fine-Tuning LLMs

6.1 Methods and Evaluation

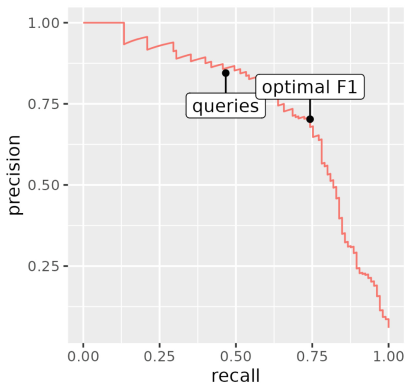
Supervised Prediction. Due to their low prevalence in the corpus, training an automatic system to detect individual argumentative fragments is challenging. The straightforward state-of-the-art approach is to fine-tune a large language model (LLM) on the manually annotated gold standard. The `combined` data set, comprising `pre` and `post`, cf. Section 3.2, consists of 1758 annotated tweets with 105 positive examples of *desire* (i.e. a prevalence of ca. 6%). Using 70% as training data and 30% as test data leaves us with 73 positive training examples (out of 1231 training examples), and 32 positive test examples (out of 527 test examples).[11]

In a first attempt, we use distilbert-base-uncased [29] as a base model and fine-tune using the `transformers` package with standard settings.[12] The trained model yields scores for both classes (*desire* and *no desire*); we focus on the positive class here. Note that scores for the two classes have a near-perfect negative correlation. A cut-off value for this score determines the trade-off between precision and recall; see Fig. 3 for the resulting precision-recall curves. A standard composite measure is the area under this curve (PR-AUC).[13] As can be seen from Fig. 3b (line *LLM (combined)*), the trained model performs poorly: precision values of, say, 50% can only be achieved with less than 25% recall (and vice

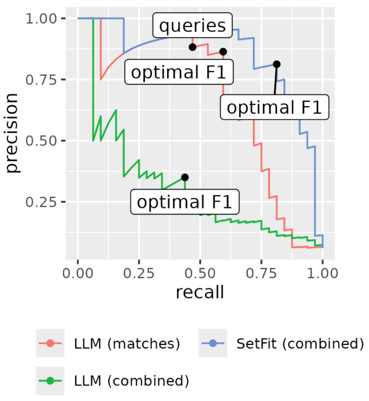
[11] All train/test splits are stratified random samples, i.e. we take random samples but make sure that the ratio of positive examples remains the same across splits.

[12] AutoModelForSequenceClassification, learning rate 2e-5, 5 epochs, 0.01 weight decay.

[13] Alternatively, we could look at the area under the receiver-operating characteristic (ROC) curve, which plots the true positive rate (precision) against the false positive rate (1 − specificity). This curve is however more suitable for situations where both classes (positive and negative) are equally prevalent or at least equally important.



(a) PR curve of LLM (trained on matches) on all of combined



(b) PR curve of different LLMs on test-split of combined.

Fig. 3. PR curves of LLMs on `combined` and its test split.

versa). The queries, on the other hand, achieve the expected high precision of 88% on the test set, and a stable trade-off with 47% recall.

As mentioned in the introduction, recent advancements in NLP have brought forth LLM frameworks that can generalise from very small numbers of training examples. Here, we use SetFit as such a few-shot classifier, and fine-tune the paraphrase-multilingual-mpnet-base-v2 model on our training examples. The PR curve of this approach outperforms the first attempt by a large margin (see Fig. 3b, line *SetFit (combined)*) and achieves competitive results compared to our query-based approach.

Generalizing from Query Matches. Additionally, we present an approach that leverages our corpus queries as training data for fine-tuning the LLM. We use all query matches, except for those in the `combined` gold standard to ensure comparability. We take 70% of a total of 145,699 matches for the *desire* pattern as positive training examples and add the same amount of random tweets (excluding query matches and those in `combined`) as negative examples. Note that for training, we assume all query matches to be instances of *desire* and randomly selected tweets to be negative examples. This is a reasonable approximation due to the high precision of the queries (ca. 96%) and the low prevalence of *desire* (ca. 6%).

Our approach can likely be improved considerably by optimizing any of the following parameters: Firstly, we could train on all query results. However, with the setting at hand, we can also evaluate how well the LLM predicts query results (see below for results). Secondly, we could provide a dataset with the (estimated) prevalence of *desire*. Lastly, we could try different base models and parameter settings (learning rate, weight decay, etc.). However, our goal here is a proof of concept, not engineering the best possible system.

Unsurprisingly, an LLM trained on query matches can accurately distinguish query matches from other tweets, i.e. it can learn the formal linguistic structures expressed by the queries. Recall that we only used 70% of the matches as positive training examples. Evaluating the LLM classifier on the remaining examples (mixed with random tweets) yields a PR-AUC of 0.9978. However, we are interested in its performance to detect the *desire* pattern, not just desire that is also captured by the queries (whose estimated recall is ca. 43%).

The precision-recall curve of this LLM on `combined` in Fig. 3a is thus more interesting. We also indicate the performance of the corpus queries in the graph. It is no coincidence that this data point lies on the PR curve of the LLM, which retrievs query matches nearly perfectly. At this point of the curve, the query matches and LLM predictions are almost identical. By lowering the LLM decision threshold, we move down the PR curve, improving recall at the cost of precision. Alternatively, we can further improve precision if we accept an even lower recall. Many reasonable trade-off points between precision and recall are available. In the graph, we also the trade-off that maximises F_1, i.e. the harmonic mean between precision and recall. We determine this value *ex post* for reasons of simplicity; in practical applications, it can also be determined on a separate development set.

Table 4. Comparison of different approaches to detect *desire* on the complete data set `combined` (top) and on test-split of `combined` (bottom). The query approach yields highest precision, the LLM trained on query matches can yield higher recall with with still decent precision values (as exemplified by the point of optimal F_1, indicated in bold).

data set	prev	approach	FN	FP	TN	TP	precision	recall	F_1
`combined`	0.06	LLM (matches)	28	33	1620	77	0.70	**0.73**	**0.72**
		queries	56	9	1644	49	**0.84**	0.47	0.60
test-split	0.06	LLM (matches)	9	6	489	23	0.79	0.72	0.75
		LLM (combined)	19	26	469	13	0.33	0.41	0.37
		SetFit (combined)	7	6	489	25	0.81	**0.78**	**0.79**
		queries	17	2	493	15	**0.88**	0.47	0.61

Figure 3b shows PR curves on the test set of `combined`, where we can compare the LLM trained on the train-split of `combined` (*LLM (random2000)*) with the one trained on query matches (*LLM (matches)*). The LLM trained on query matches is far superior to the one trained only on a couple of dozen of positive examples. Table 4 lists detailed results for all approaches on `combined` and its test-split (for LLMs, the decision threshold is set at the point of optimal F_1). In terms of precision, the corpus queries yield the best results (as by design).

However, the LLM trained on query matches can yield better recall, as is exemplified at the point of optimal F_1 of the PR curve.

6.2 Discussion: Qualitative Comparison of Approaches

As seen in Table 4, at the point of optimal F_1, the LLM approach trained on query matches achieves higher recall than the queries, but lower precision. In this section, we examine the differences between these two approaches through a qualitative analysis of tweets in the gold standards that were found by the LLM, but not by the queries.

The first group of tweets are newly identified true positives, i.e. tweets that contribute to the LLM achieving higher recall than the queries. The results suggest that the improvement in recall can be attributed to systematic factors. The main patterns in the true positives unique to the LLM include tweets containing typos (*Britian*) or short modifier phrases (*Denmark* ***for one*** *will be queuing up to leave*). While it would technically be possible to write queries that handle such cases, such modifications would either introduce unwarranted complexity for relatively small improvements, or they would reduce the queries' precision by introducing more opportunities for false positives.

Additionally, the queries rely on linguistic pre-processing, in particular on POS tagging. While this information is helpful in specifying grammatical patterns, tagging errors occasionally prevented the queries from finding relevant tweets. Thus, the LLM found several nominalizations that the POS tagger misinterpreted as adjectives, causing the query to miss a noun phrase (e.g. *The British want EU migrants to stay*). Similarly, the queries impose semantic restrictions via word lists where necessary, which obviously limits the scope of words that can possibly be matched in a given position. In contrast, the LLM found tweets with unusual entities like *noted Europhile paper backs Brexit*.

Finally, some hits found by the LLM contained syntactic patterns for which we had no queries – either because the expression contained non-standard syntax (*If we Brexit., ending the Barnet agreement, I'm for!*), or because the constructions were too rare to reasonably justify developing a manual query (*Very much looking forward to seeing nigel farage in action tonight*).

False positives (FP) unique to the LLM were usually syntactically similar to one of the queries, but did not match the correct semantics (*#Brexit gloom is for losers*). In rarer cases, the tweets contained some reference to desire that was too implicit according to the guidelines (*"Being pro brexit is wacist!" said the hipster white brits to the black brits* – this tweet is not considered *desire* since it is a general statement rather than a specific entity desiring something).

7 Limitations

The case study currently pursues a comparatively narrow topical focus; the generalizability of our findings remains to be explored. Scaling the overall approach to large repositories of logical patterns is possible in principle but resource-intensive: Firstly, the method relies on the manual development of corpus queries, which involves corpus-linguistic analysis, and secondly, query development needs manual annotation of random samples to find suitable starting points (however, queries then generalise from very few examples). The task of annotating samples for matches with logical patterns is conceptually difficult, and agreement between human annotators is comparatively low (with notably higher agreement on the `post` dataset). Our approach to fine-tuning LLMs using query results is currently at the proof-of-concept stage and could likely be substantially improved in further work.

8 Conclusion

We have described an approach to extracting argument fragments from short text snippets on social media, using corpus queries to fill slots in predefined logical patterns. Patterns and queries can be applied in a nested fashion, allowing for the extraction of more complex semantic content. We have demonstrated an application of our methodology in the generation of training data for use in the fine-tuning of LLMs. Without any manual annotation, we achieve comparable results to state-of-the-art few-shot learning approaches such as SetFit that have been trained on more than 1200 manually annotated tweets.

Ongoing efforts aim to conduct automated logical reasoning steps over the extracted argument fragments, which will require use of semantic similarity measures. Moreover, we intend to extend the scope of the method both w.r.t. supported lanuages and w.r.t. the length and degree of coherence of the underlying text, covering also, e.g., newspaper articles or parliamentary debates, and aiming to extract argumentation chains instead of just argument fragments.

Acknowledgements. The work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 377333057 "Reconstructing Arguments from Newsworthy Debates (RAND)".

References

1. Alsinet, T., Argelich, J., Béjar, R., Cemeli, J.: A distributed argumentation algorithm for mining consistent opinions in weighted Twitter discussions. Soft. Comput. **23**(7), 2147–2166 (2019). https://doi.org/10.1007/s00500-018-3380-x
2. Beck, T., Lee, J.U., Viehmann, C., Maurer, M., Quiring, O. and Gurevych, I.: Investigating label suggestions for opinion mining in german covid-19 social media (2021)
3. Bhatti, M.M.A., Ahmad, A.S., Park, J.: Argument Mining on Twitter: a case study on the planned parenthood debate. In: Proceedings of the 8th Workshop on Argument Mining, pp. 1–11. Association for Computational Linguistics, Punta Cana, Dominican Republic (2021) https://doi.org/10.18653/v1/2021.argmining-1.1 https://doi.org/10.18653/v1/2021.argmining-1.1

4. Bosc, T., Cabrio, E., Villata, S.: Tweeties squabbling: positive and negative results in applying argument mining on social media. In: Computational Models of Argument, COMMA 2016. Frontiers Artificial Intelligence Applications, pp. 21-32. IOS Press (2016)
5. Cabrio, E., Villata, S.: Five years of argument mining: a Data–driven Analysis. In: International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 5427-5433. ijcai.org (2018)
6. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: an Architecture for Development of Robust HLT applications. In: Annual Meeting of the Association for Computational Linguistics, ACL 2002, pp. 168-175 (2002). https://doi.org/10.3115/1073083.1073112
7. Dusmanu, M., Cabrio, E., Villata, S.: Argument mining on Twitter: arguments, facts and sources. In: Empirical Methods in Natural Language Processing, EMNLP 2017, pp. 2317-2322. ACL (2017)
8. Dykes, N., Evert, S., Göttlinger, M., Heinrich, P., Schröder, L.: Argument parsing via corpus queries. Inf. Technol. **63**(1), 31–44 (2021). https://doi.org/10.1515/itit-2020-0051
9. Dykes, N., Evert, S., Göttlinger, M., Heinrich, P., Schröder, L.: Reconstructing arguments from noisy text: introduction to the RANT project. Datenbank- Spektrum **20**, 123–129 (2020)
10. Evert, S., Hardie, A.: Twenty-first century Corpus Workbench: updating a query architecture for the new millennium. In: Corpus Linguistics, CL 2011. University of Birmingham (2011)
11. Evert, S.: The CWB development team: the IMS Open Corpus Workbench (CWB) CQP Interface and Query Language Tutorial. CWB Version 3.5. 2022. https://cwb.sourceforge.io/documentation.php
12. Feger, M., Dietze, S.: TACO–Twitter Arguments from COnversations. (2024)
13. Feng, S.Y., et al.: A Survey of data augmentation approaches for NLP. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 968-988. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.findings-acl.84
14. Goudas, T., Louizos, C., Petasis, G., Karkaletsis, V.: Argument extraction from News, blogs, and Social Media. In: Artificial Intelligence: Methods and Applications, SETN 2014, pp. 287-299. Springer (2014)
15. Grosse, K., Chesñevar, C., Maguitman, A., Estevez, E.: Empowering an eGovernment platform through Twitter-based arguments. Inteligencia Artif. **15**(50), 46–56 (2012)
16. Hardie, A.: CQPweb - combining power, flexibility and usability in a corpus analysis tool. Int. J. Corpus Ling. **17**(3), 380–409 (2012)
17. Humml, M., Schröder, L.: Common Knowledge of abstract groups. In: AAAI Conference on Artificial Intelligence (AAAI 2023), pp. 6434-6441 (2023). https://doi.org/10.1609/aaai.v37i5.25791
18. Liu, H., et al.: Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. arXiv preprint arXiv:2205.05638 (2022). https://doi.org/10.48550/arXiv.2205.05638
19. Lytos, A., Lagkas, T., Sarigiannidis, P., Bontcheva, K.: The evolution of argumentation mining: from models to social media and emerging tools. Inf. Process. Manage. **56**(6), 102055 (2019). https://doi.org/10.1016/j.ipm.2019.102055
20. Lytos, A., Lagkas, T., Sarigiannidis, P.G., Argyriou, V., Eleftherakis, G.: Modelling argumentation in short text: a case of social media debate. Simul. Model. Pract. Theory **115**, 102446 (2022). https://doi.org/10.1016/J.SIMPAT.2021.102446

21. Minnen, G., Carroll, J., Pearce, D.: Applied morphological processing of English. Nat. Lang. Eng. **7**(3), 207–223 (2001)
22. Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., Smith, N.: Improved part-of-speech tagging for online conversational text with word clusters. In: Human Language Technologies, HLT-NAACL 2013, pp. 380-390. ACL (2013)
23. Pantel, P., Pennacchiotti, M.: Espresso: leveraging generic patterns for automatically harvesting semantic relations. In: Computational Linguistics / Annual Meeting of the Association for Computational Linguistics, ACL 2006. ACL (2006)
24. Proisl, T., Uhrig, P.: SoMaJo: state-of-the-art tokenization for German web and social media texts. In: Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task, pp. 57-62. Association for Computational Linguistics, Berlin (2016). https://doi.org/10.18653/v1/W16-2607
25. Qiu, Y., Jin, Y.: ChatGPT and finetuned BERT: a comparative study for developing intelligent design support systems. Intell. Syst. Appl. **21**, 200308 (2024). https://doi.org/10.1016/j.iswa.2023.200308
26. Rahman, A.M.M., Yin, W., Wang, G.: Data augmentation for text classification with EASE. In: Abbas, M., Freihat, A.A. (eds.) Proceedings of the 6th International Conference on Natural Language and Speech Processing (ICNLSP 2023), pp. 324-332. Association for Computational Linguistics, Online (2023)
27. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using siamese BERT-Networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2019)
28. Ritter, A., Mausam, Etzioni, O., Clark, S.: Open domain event extraction from twitter. In: Knowledge Discovery and Data Mining, KDD 2012, pp. 1104- 1112. ACM (2012)
29. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR abs/1910.01108 (2019)
30. Schaefer, R., Stede, M.: Argument mining on Twitter: a survey. Inf. Technol. **63**(1), 45–58 (2021). https://doi.org/10.1515/itit-2020-0053
31. Shnarch, E.,et al.: Will it Blend? blending weak and strong labeled data in a neural network for argumentation mining. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 599-605. Association for Computational Linguistics, Melbourne, Australia (2018). https://doi.org/10.18653/v1/P18-2095
32. Son, Y., et al.: Recognizing counterfactual thinking in social media texts. In: Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (2017). https://doi.org/10.18653/v1/p17-2103
33. Tunstall, L., et al.: Efficient few-shot learning without prompts. arXiv preprint arXiv:2209.11055 (2022). https://doi.org/10.48550/ARXIV.2209.11055
34. Walton, D., Reed, C., Macagno, F.: Argumentation Schemes. Cambridge University Press (2008). https://doi.org/10.1017/CBO9780511802034