Meisenbacher, Stephen; Norlander, Peter

Working Paper

Creating Data from Unstructured Text with Context Rule Assisted Machine Learning (CRAML)

GLO Discussion Paper, No. 1214

Provided in Cooperation with:
Global Labor Organization (GLO)

Suggested Citation: Meisenbacher, Stephen; Norlander, Peter (2022) : Creating Data from Unstructured Text with Context Rule Assisted Machine Learning (CRAML), GLO Discussion Paper, No. 1214, Global Labor Organization (GLO), Essen

This Version is available at:
http://hdl.handle.net/10419/267553

Creating Data from Unstructured Text with Context Rule Assisted Machine Learning (CRAML)

Stephen Meisenbacher[1,2]

Peter Norlander[1,3]

December 19, 2022

Abstract. Popular approaches to building data from unstructured text come with limitations, such as scalability, interpretability, replicability, and real-world applicability. These can be overcome with Context Rule Assisted Machine Learning (CRAML), a method and no-code suite of software tools that builds structured, labeled datasets which are accurate and reproducible. CRAML enables domain experts to access uncommon constructs within a document corpus in a low-resource, transparent, and flexible manner. CRAML produces document-level datasets for quantitative research and makes qualitative classification schemes scalable over large volumes of text. We demonstrate that the method is useful for bibliographic analysis, transparent analysis of proprietary data, and expert classification of any documents with any scheme. To demonstrate this process for building data from text with Machine Learning, we publish open-source resources: the software, a new public document corpus, and a replicable analysis to build an interpretable classifier of suspected "no poach" clauses in franchise documents.

Keywords: machine learning, natural language processing, text classification, big data

JEL classification: B41, C38, C81, C88, J08, J41, J42, J47, J53, and Z13

---

[1] Equally contributing authors. Correspondence can be sent to stephen.meisenbacher@tum.de and pnorlander@luc.edu. Under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License, we release: CRAML software at https://github.com/sjmeis/CRAML_Beta/, machine learning classifiers for job advertisement text at https://zenodo.org/record/7454652, and the replication materials for the no poach analysis at https://zenodo.org/record/7454758.

We are grateful for support from the Economic Security Project Anti-Monopoly Fund; Loyola Rule of Law Institute; Loyola Quinlan School of Business; and Loyola University Chicago. Work with job advertisement text has been supported (in part) by Grant # 2111-34962 from the Russell Sage Foundation. We thank: Patricia Tabarani, Chloe Clark, Zach Nelson, and Damian Orozco for research assistance; Laura Zbella, Angelica Vaca, Denise DuVernay, and Joseph Koral for research support; DocumentCloud, Michael Morisy, and Mitchell Kotler for replication support; Eric George, Steve Sauerwald, and Chris Erickson for comments on earlier drafts; participants at the 86th Midwest Economic Association, the 74th Annual Labor and Employment Relations Association, 42nd Annual Strategic Management Society, and 82nd Academy of Management conferences. Errors are ours.

[2] Technical University of Munich, School of Computation, Information and Technology

[3] Loyola University Chicago, Quinlan School of Business; Fellow, Global Labor Organization.

Creating Data from Unstructured Text with Context Rule Assisted Machine Learning (CRAML)

Advances in computational methods offer new ways to gain insight from large volumes of unstructured text, but have significant limitations, tradeoffs, and a lack of clear guidelines (Pandey and Pandey 2017). Despite advanced Natural Language Processing (NLP), Machine Learning (ML), Deep Learning (DL), and Artificial Intelligence (AI), "the herculean task of finding constructs using full-text search" remains (Larsen and Bong 2016).

Systematic research and processes are needed to create structured data from unstructured texts (DiMaggio 2015) that lack a schema, are not standardized, have multiple formats, and come from diverse sources (Adnan and Akbar 2019). Zettabytes (ZB) of new data are produced daily (Begum and Nausheen 2018), and 80% is unstructured (Hammoud et al. 2019). For knowledge professionals, just-in-time access to domain specific document repositories can unlock value (Subramani et al. 2021). However, structured information is a pre-requisite for researchers and organizations performing quantitative analysis and "as much as 80% of an organization's data is 'dark"' (Lacity and Willcocks 2021). While manual expert labor required for qualitatively coding novel classification schemes is hard to scale, statistical packages, many information systems, and quantitative social scientists expect data to be structured in a particular format: "tabular data – variables in columns, cases in rows" (Lazer and Radford 2017).

To address the disjunction, we describe a methodological bridge: an expert-built set of context rules scaled to classify unstructured text and build training data for Machine Learning. Context Rule Assisted Machine Learning (CRAML) is a hybrid method for structuring textual data from large-scale corpora into datasets ready for training ML models. CRAML empowers domain experts to mine volumes of text and scale classification schemata efficiently, and returns tabular data that captures the occurrence of concepts within a corpus of documents.

This research methods paper contributes to information systems research on the design of intelligent systems that can generate economic and social value from unstructured data (Abbas et al. 2018). By clearly defining the problem, managing the training data before AI is attempted, and affording users the opportunity to evaluate and adjust both inputs and outputs, CRAML addresses major concerns identified in research on AI (Zhang et al. 2020). Such expert-built rule-assisted Machine Learning models are described in the literature and are found to outperform benchmark methods and professional perception in critical cases such as detecting persons in states of emotional distress (Chau et al. 2020). This is the first paper to present software and guidelines for advanced users of an ordinary desktop computer to build Machine Learning classifiers and rectangular datasets from unstructured text according to any scheme.

We first use CRAML to search for relevant literature in management that highlights current empirical approaches to the analysis of large text corpora. We next describe the use of CRAML to study characteristics of job advertisements, and set a new standard of transparency for research studying job advertisement text, where underlying proprietary data is obtained under license. We release ML classifiers, and the rules that created them, to enable other researchers using job advertisement text to replicate and interpret our approach. Lastly, we publish a new, large-scale document corpus of mandatory franchise disclosure documents and present an end-to-end analysis to enable a full replication of our process. We detect thousands more "no poach clauses" that could violate state and federal antitrust law than found in earlier research relying on a proprietary data provider's analysis (Krueger and Ashenfelter 2022). We release a rectangular dataset that represents the prevalence of suspected no poach clauses that restrict the ability of franchise firms to recruit employees from other companies that belong to a franchise system. We release ML classifiers built with CRAML, key information that any user can use to build their

own classifier from their own corpus, and provide additional replication data: the text extracted from franchise documents, the manually coded rules that create the structured, labeled data output, the training data, and a ML classifier that detects suspect no poach clauses.

We review the well-known challenges CRAML was created to address, provide motivation for our solution, and elaborate on the methodological detail of CRAML with an empirical analysis of franchise no poach agreements. Appendix A provides additional description for conducting a literature search in CRAML, Appendix B describes opportunities for transparency with sensitive data, and technical details are described in Appendix C.

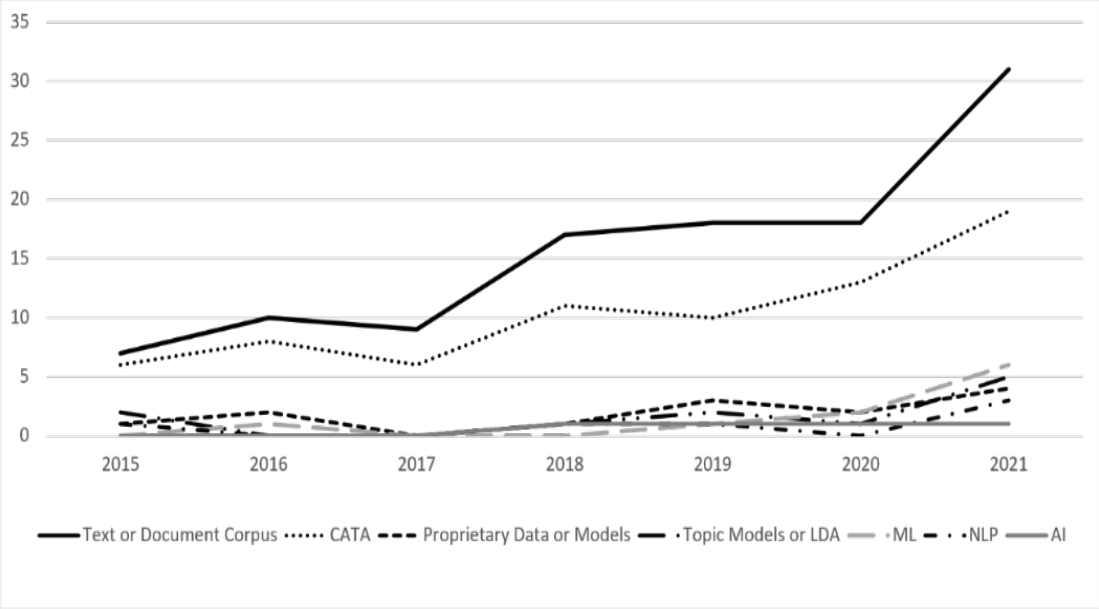
THE STATE OF THE ART

Across many fields of research, methods for analyzing unstructured text are crucial. We reviewed the literature in Management, Information Systems, Computational Social Science and Computer Science on Machine Learning (ML), Artificial Intelligence (AI), Natural Language Processing (NLP), Topic Modeling and Latent Dirichlet Allocation (LDA), Linguistic Inquiry and Word Counting (LIWC), Computationally Aided Text Analysis (CATA), and text or document corpora. In order to aid our review, we used the CRAML software to narrow our search for relevant literature within a corpus of 1,994 papers published from 2015-2021 in five top empirical management journals (Academy of Management Journal, Administrative Science Quarterly, Journal of International Business Studies, Organization Science, and Strategic Management Journal). Within the management literature corpus, we identified papers that demonstrate the use of a method for transforming unstructured text into data. Typical academic search engines would be unable to retrieve or pinpoint the relevant papers. In Appendix A, we provide additional detail on how CRAML can be used in bibliographic research.

Word Counting Approaches (LIWC, CATA)

Computer-aided text analysis (CATA) and Linguistic Inquiry and Word Count (LIWC) are two techniques to classify unstructured text. Experts develop lists of words or dictionaries that correspond to an overall construct and measure the occurrence of keywords inside specific documents (Short and Palmer 2008; Short et al. 2010). Researchers use LIWC standard dictionaries to detect common constructs such as positive / negative sentiment in tweets (Bachura et al. 2022). Researchers can also develop custom dictionaries to capture previously unstudied phenomena: for example, to measure how actors use cultural toolkits (Weber 2005). Research often begins by deductive keyword selection and then interrogation of text using n-grams (McKenny, Short, and Payne 2013). To locate papers that count words to build data from unstructured text, we developed keywords deductively. We used CRAML software to build indicators for whether each academic paper in the corpus contains keywords that fall within a given construct: we focused on a subset of 110 papers that refer to text data or a document corpus. Figure 1 shows that text corpus analysis, and word counting along with it, have grown over time: 88 of the 110 papers that mention a text corpus also refer to CATA.

Figure 1: Counting Words: Management Papers with Text Data and CATA, AI, ML, LDA, NLP



Limitations of Keyword and Word-Counting Approaches

Conceptual schemes that rely only on keywords are not very rich (Duriau, Reger, and Pfarrer 2007) and may not be well-captured by word frequency approaches either. These methods perform less well than ML at tasks such as inferring personality (Cutler et al. 2021). Word counting assumes that all word occurrences equally contribute to the measurement of a construct - but every mention of "artificial intelligence" captures both papers that use an AI method, as well as those that mention AI phenomena in passing. Because the method relies on exact matching and correct spellings, word matching will fail to recognize concepts when faced with messy data. Without the context surrounding the keywords, meaning and accuracy are lost. When describing the frequency of terms inside a document, CATA methods do not typically adjust for document length, although additional steps can be taken (Guo, Sengul, and Yu 2021). While keywords and word-counts might start a search for relevant concepts in literature, qualitative researchers who seek to inductively build theory from corpora typically must also engage in qualitative hand coding (Li, Zhang, and Kettinger 2022).

Unsupervised Learning (Topic Modeling and LDA)

Text clustering aims to solve the problem of information overload by simplifying a mass of text into a smaller number of categories. While many text clustering methods exist (Aggarwal and Zhai 2012), a popular technique is *topic modeling*, which aims to learn "topics" from an unstructured collection of text documents. Data is often first pre-processed (stemmed), stripped of common words, and converted into a document-term matrix, which leaves only the stems of latent keywords for analysis. Latent Dirichlet Allocation (LDA) then assigns the words in a document to a topic, under the assumption that every part of every document contributes to some topic. LDA's versatility is that it is not domain-specific: it can aid inductive theorizing (Shrestha

et al. 2021), study hacker forums (Yue, Wang, and Hui 2019), why people retweet (Geva, Oestreicher-Singer, and Saar-Tsechansky 2019), online communities (Bapna, Benner, and Qiu 2019), and discourse on emerging technologies like blockchain (Miranda, Wang, and Tian 2022).

Of the 110 text corpus papers in the management literature, 11 papers discuss topic models, with five of these papers appearing in 2021. This method has been used to study how firms interact with the legal environment based upon annual reports (Giorgi, Maoret, and Zajac 2019), and to arrive at text-based measures of corporate dissimilarity from corporate annual reports, requiring 25 crowd-sourced MTurkers to ensure themes are reliably identified for the top 25 of 125 latent topics chosen (Choi, Menon, and Tabakovic 2021). LDA has been used to study patent text and recombination in breakthrough innovation (Kaplan and Vakili 2014), abstracts within a citation network (Sine, Cordero, and Coles 2022), and CEO speech in combination with ML analysis of CEO facial expressions (Choudhury, Allen, and Endres 2020).

Limitation: Ambiguous Output

LDA assigns all text in every document inside a text corpus to a theme, often generating output that is difficult to interpret without additional data. Papers reviewed here address this by limiting the scope of their corpus and extensively pre-processing to remove potential noise (also removing potentially relevant context and data) and post-hoc interpretive analysis with expert or crowd-sourced judgement. Topic modeling assumes that one must know the number of topics that exist within a collection of documents. This is non-trivial, particularly with massive corpora. Finally, topic modeling imposes an assumption that a researcher be interested in an “unbiased” representation. Experts often disagree about the interpretation of text, and a quest for unbiased (as opposed to transparent and replicable) analysis is misguided (Nelson 2019). Ultimately,

unsupervised computational methods are largely unable to produce rich results in the absence of human involvement (Nelson 2017).

The Modern ML Paradigm

Supervised Machine Learning models are widely used in industry: to evaluate candidates for employment (Cohen and Mahabadi 2022), support managers in giving employee feedback (Tong et al. 2021), moderate text content on the internet (Gorwa, Binns, and Katzenbach 2020; Gillespie 2020), judge the sentiment of customer-written reviews (Liu, Li, and Xu 2021), and build proprietary datasets for financial and academic use. However, ML models of text are often uninterpretable, secret, and may be perceived as too difficult for users with low computational resources to access. Among 10 papers in the management corpus that discuss ML, and four more that discuss AI, we could not locate any that describe the development of a new academic text classification model that is used to construct new data from unstructured text.

While general AI and ML methods rapidly advance, there is a great need for applications in specific domains, such as detecting banking fraud (Abbasi et al. 2012) or fake websites (Abbasi et al. 2010). "Human-in-the-Loop" or hybrid systems are increasingly prominent, introduce responsibility into intelligent agents, and increase interpretability (Zanzotto 2019). Likewise, human-in-the-loop for ML and NLP is particularly promising, both for data pre-processing and model learning (Wu et al. 2022). Efforts to use AI and exclude domain experts are likely to fail, as shown in a study of AI use in hiring decisions and hybrid practices (Broek, Sergeeva, and Huysman 2021). AI systems cannot (and should not) be trained without human oversight, and thus procedures to incorporate human knowledge into these systems is vital.

For supervised ML, challenges lie not so much with the ML methods or models themselves, but with the paradigm for ML. What we call "the Modern ML Paradigm" for supervised learning

of text data largely sees training data only as input, and holds the interpretability and explainability of ML model outputs as a post-hoc concern. Much of the attention around Explainable AI (XAI) has been placed on "the way in which models behave," seeking meaning and interpretability in the outputs of models that can often be highly opaque (Adadi and Berrada 2018). However, labeled training data is the critical input to supervised ML models to code future, unseen text instances. If building custom training datasets for text were easier, then resulting models could be interpreted – and modified – from the onset, i.e., at the input stage. Expert-trained ML models could be interpretable and put into action in a domain, and escape some of the limitations of the modern ML paradigm: black boxes, proprietary training data, and un-reproducible, meaningless output that exclude participation from those without computational power and advanced computer science knowledge.

Limitation: The Black Box of Supervised Learning

The "Black Box" refers to the mystery by which textual inputs are categorized by machines leading to certain outputs and are rooted in increasingly large model architectures and consequently, how computational models *memorize*, rather than truly *learn*. In turn, one is often hard pressed to decipher the behavior of a classifier, even with state-of-the-art models – they simply reflect patterns observed during training. Trust requires understanding "what the machines are doing" (Castelvecchi 2016). In contrast, the modern ML paradigm attempts to "explain the black box" after the fact. Post-hoc approaches have become the *de facto* way of achieving some level of explainability with complex DL models (Arrieta et al. 2019).

Without understanding the data that goes into training the model, the explainability of the modern ML paradigm is limited. A recent survey of explainability in supervised ML methods calls for rethinking the problem from first principles, and encourages researchers to ask "What

are we actually looking for? Do we really need a black box model?" (Burkart and Huber 2021). For many experts, the immediate answer is no. Even a model with a high degree of predictive power is undesirable if it does not produce interpretable results (Nelson 2017, 2019).

Underspecification can result from training models on massive volumes of data and selecting the model(s) that achieve a desired predictive power (D'Amour et al. 2022). Underspecification often leads to failure when those models are used outside of the context in which they were trained: "the process used to build most machine-learning models today cannot tell which models will work in the real world and which ones won't" (Heaven 2020). Explainability when models are faced with new situations would be a "sign of mastery" (Gunning et al. 2019).

Limitation: Proprietary Training Data and Methods

Seen from a different angle, black boxes may protect trade secrets hidden behind such trained models (Rudin 2019). Secrecy in cases where "high-stakes" decisions are made by the models in question have potentially significant societal or economic implications. Academic users often rely on commercial data providers that give little to no insight into their proprietary process for data creation. In our analysis of the management literature, 13 of the 100 text analysis papers, and 117 papers in total indicate the use of licensed, proprietary data or models. The problem identified by Lazer et al. (2009) is that computational methods could become the "exclusive domain" of private companies and government agencies that operate contrary to the academic commitment to openness. The inadequacy of data-sharing paradigms for big data cast doubt regarding the veracity of ML models (Lazer et al. 2020).

Private firms and academics advance the state-of-the-art by breaking new records for the complexity and the size and number of parameters in a model. Potential contributions from researchers who lack access to significant computing resources, the funds to buy proprietary

datasets, or access to confidential data are excluded (Card et al. 2010). When big or secret data becomes the criteria through which research is judged, only those with access can compete, and there may be less attention to "cumulative progress toward answering important questions" (Davis 2015).

By reducing the need for humans to think, act, or belong (contribute, be included) in the public sphere, ML systems can limit opportunity (Kane et al. 2020). To deliver on the potential of ML, improvements needed include greater sharing of data, protections for privacy to enable more open data, intellectual property standards to reduce transaction costs, and approaches that enable genuine replication (King 2011). Pre-processing of datasets for supervised learning could be a core strength and contribution of social scientists (DiMaggio 2015). Calls for greater systematic research to develop solutions for curation challenges and guidelines for pre-analysis and developing training data have largely been unmet (ibid).

Limitation: Meaningful Contributions to Knowledge

In a statement that rings true almost a decade after it was written, there is little evidence of computational social science in leading social science field journals (Watts 2013), or in the management corpus reviewed here. Computational social science methods have advanced significantly, but it has yet to be demonstrated how computational scientific infrastructure can be applied to important social problems (Lazer et al. 2020). Disciplinary silos, privacy concerns, and unreliable, non-replicable, and proprietary data hamper the potential for computer science methods to provide meaningful insight into society's challenges (Lazer et al. 2020).

The emergence of a computational social science speaks to the hope that ML and AI can advance knowledge and be impactful within academia. However, the possibilities are lessened when such tools are perceived as beyond the reach of ordinary users (Faik, Barrett, and Oborn

2020). Initiatives such as "data science for good" have been criticized for not giving the social sector "the opportunity to design for what we want" (Porway 2022). Without tools that enable domain experts to develop their own ML models, there is a risk that new interdisciplinary centers might lead to yet another siloed academic discipline as ill-equipped to solve problems as others (Abbott 2001). Future opportunities for AI include democratization, reducing requirements for data, and enhancing AI explainability and transparency (Benbya, Davenport, and Pachidi 2020).

Current ML research has lost "its connection to problems of import to the larger world of science and society" (Wagstaff 2012). The (over)use of benchmark datasets, many of which are "synthetic", has "glaring limitations" and still make reproducibility difficult. Only 1% of ML papers are applied to a specific domain and the rest use benchmark datasets (ibid).[1] By way of contrast, 90% of research constructs are uncommon, suggesting that using pre-labeled standard benchmark data for training will miss entire vast areas of domain-specific knowledge (Larsen and Bong 2016). The focus on "abstract metrics" like F-scores to determine model quality across domains is a "mirage" that masks the need to focus on impact and usefulness of a model (Wagstaff 2012). A final issue is the "lack of follow-through" (ibid), suggesting that current computational research leaves little incentive for connection to sustained research agendas.

A related concern is the rapidly depreciating relevance of data that many ML models are trained on (Lazer et al. 2020). In the ML training process, models are trained "in the lab" on a training dataset, yet the environment in which models are deployed is dynamic. "Big data" research may apply only to specific users of a specific portal at a specific time in which a study is conducted (Hargittai 2015; Pfeffer et al. 2022). For that reason, online studies and experiments

[1] Benchmark datasets include the Movie Review Dataset (Maas et al. 2011) or the Yelp Review Dataset. While use of standardized datasets may reduce the reproducibility crisis in ML (Kapoor and Narayanan 2022), there is a tradeoff: standardized datasets have limited application to the real world.

with large response rates may fail to produce reliable or general knowledge (Bradley et al. 2021). A static training process may simply not reflect the real-world environment or its rapid evolution (Babic et al. 2020). The velocity of internet-based data also creates a limitation for computational social scientists' conclusions (Munger 2019). Solutions to these obstacles include forming new multi-disciplinary, international journals that permit rapid publication of rigorous quantitative descriptions of rapidly evolving phenomena (Munger, Guess, and Hargittai 2021).

Seeking Explainable Tools for Social Science

To sum up the issues described in this section, each method reviewed has strengths, limitations, and tradeoffs for experts who seek to analyze unstructured text. For researchers, the methods reviewed so far do not allow for expert-led, targeted extraction of niche constructs within corpora and guided construction of standardized data. For many qualitative researchers, the systematic analysis and discovery of new patterns within domain-specific digitized records could be a core contribution – the ability to find the diamond (a single topic) in the rough (lots of text). If qualitative hand-coded contributions could be easily scaled, it can address some issues found in the supervised ML domain.

CONTEXT RULE ASSISTED MACHINE LEARNING (CRAML)

Many of the problems identified with the modern ML paradigm, and the limitations of unsupervised classification and word counting methods, are addressable if accountable experts create training data for niche classifiers to detect constructs that are relevant within a specific domain. The CRAML method addresses each of six dimensions that explain the performance of AI systems: CRAML models are specific, the goals must be clear, the training and input data are context-specific, the output data is interpretable, and the environment is domain specific and expert-guided (Asatiani et al. 2020). CRAML gives qualitative and quantitative researchers new

capabilities to steer analysis and measurement of phenomena found in massive volumes of unstructured text.

A Framework for Unstructured Text Analysis

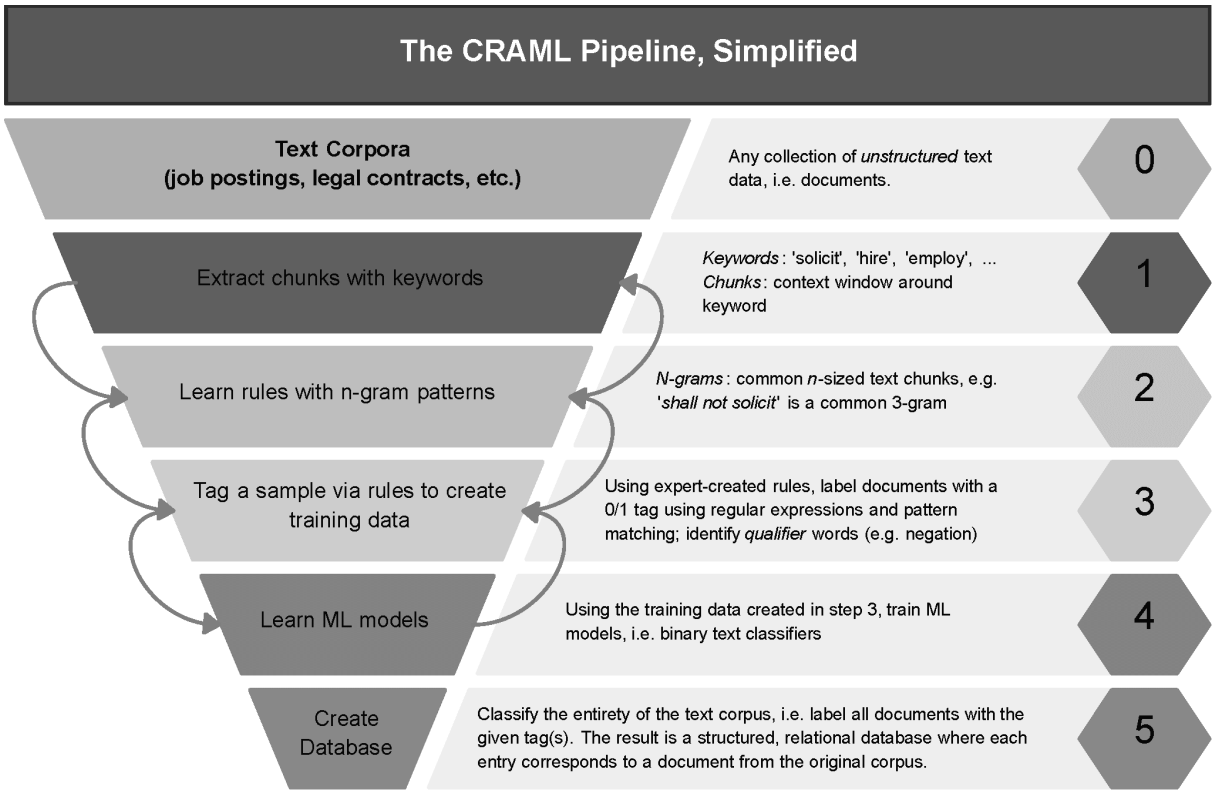
CRAML software gives users tools to (1) formulate keywords to extract relevant data from a sample of a documents in a corpus, (2) create "tags" that capture topics or constructs, and subsequently, (3) develop "context rule" sets that codify knowledge about "chunks" of text that correspond to specific tags. In other words, CRAML steps 1-3 shape training data for (4) building a niche Machine Learning text classification model. CRAML yields (5) tabular output that matches document-level metadata with a 0/1 indicator for each tag in a document. Figure 2 provides an overview of the CRAML framework.

CRAML begins with the full text of a corpus of documents (Step 0). While metadata is not required, the software is designed to combine characteristics extracted from the text with metadata at the document or record ID level. In step 1, the researcher chooses keywords to extract from the corpus, and retrieves a "chunk": a block of text containing a user-selected number of words surrounding the keyword. The chunk length should contain the full "context window": a span of the relevant context for a human to determine if the keyword-containing chunk is relevant or not for some binary classification schema. The user can extract all chunks from all of the documents in the corpus, or in low-resource environments, can randomly sample from the documents in order to perform Steps 2-4 on a smaller subset.

In step 2, the researcher analyzes the patterns within the chunks, studying n-grams that reveal the most common phrasings surrounding the keyword. In step 3, the researcher writes "context rules" that elaborate an ever-more detailed scheme of binary classifications of the chunks. We refer to these binary classification schemata as "tags": each tag represents a topic or theme that

the researcher defines by recognizing patterns in the extracted text and determining if each chunk is (tag=1), or is not (tag=0), illustrative of the tag. There is no limit on the number of tags a researcher can define. For each tag, there is no limit to the number of rules that can be defined using either exact text matching or regular expressions (RegEx).

Figure 2: A Bird's Eye View of the CRAML Process



In step 4, the researcher written context rules are applied or "extrapolated" over either the full extracted chunks or a sample of the extracted chunks to create a training dataset for ML modeling. Extrapolation labels each chunk with a 0 or 1 for each "tag," and creates a dataset that can be used for training ML models. In step 5, models learned on the training dataset using common ML algorithms tag the full extracted chunks, and the researcher identifies the best performing model. In the final step, the results are aggregated into a document-level database of structured information: for any document within the text corpus, the researcher knows if the

document contains a tag. Additional hand-coding by independent third-parties can validate that the training dataset extrapolation, and the ML results, are highly accurate.

CRAML's pipeline incorporates a versatile set of tools, and researchers may appreciate its flexibility and multiple uses: a researcher may wish to pursue analysis using only pieces of the CRAML process. As Pandey and Pandey (2017) note, choosing between a dictionary, rules, or supervised Machine Learning involves tradeoffs, and no clear guidelines. In CRAML, particular steps and technologies can be interchanged according to the user's preference, skill set, and research challenge. With relatively low hardware requirements, CRAML seeks to eliminate barriers to accessing ML technology (Hedderich et al. 2021). A qualitative researcher can quickly sample and sift through text data and experiment with alternative classification schemes and definitions. For another researcher, extraction could be a pre-processing step before engaging in a computational grounded theory project (Nelson 2017) that uses unstructured NLP methods once the plausibly relevant themes are extracted based upon keywords. A third researcher may wish to extrapolate a full dataset without ML. A fourth researcher may wish only to sample a massive corpus, extract context windows around keywords, and inductively generate topics or themes with extracted text.

A Bridge to Solving Real-World Problems

By positioning context rules as the bridge between domain expert knowledge and structured datasets that represent embodied knowledge, CRAML provides a "traceability" from datasets classified by ML models back to the context rules that lead to them – and the humans who wrote those. ML modeling "presumes a world of already existing (divine or rationalistic) rules, which only need to be formalized, in order to make sense to a machine (or an analytical philosopher for this reason)" (Apprich 2018). By focusing on explainability at the first bridge from unstructured

data to structured data(sets), the second bridge from training data to trained models becomes easier to cross. Context rules in CRAML codify which text belongs to which topic or theme according to the user. To the extent such rules can be articulated consistently, they can be extrapolated over unstructured text to create fully structured datasets to train ML models.

Rather than presuming unbiased training data, CRAML extends the user's worldview to the realm of ML models via context rules. Rigorous manual content analysis by qualitative social scientists is often geared toward the induction of topics and themes, yielding new constructs, analytical frameworks, or mid-level theory of some phenomenon (Glaser and Strauss 1967). New methods for computational grounded theory (Nelson 2017) and scaling methods such as coding over large volumes of text can contribute to theory development (Tchalian 2019). A quantitative or hypothesis-driven researcher might pursue evidence of a phenomena within a text corpus, quantify its frequency, and then analyze it. Through the extrapolation process, CRAML requires manual work only in the crucial first stages, and automates the rest at scale. CRAML's initial step is similar to processes used in IS research: for example, a specialized ontology of hypotheses built with rule-based approaches for extraction and word embeddings (Li, Larsen, and Abbasi 2020). The "human in the loop" contributes transparent manual codifications of text, which allow for the replicable and configurable construction of tabular datasets used either immediately for quantitative social science analysis, or to train ML models. The ML model that results from CRAML is interpretable as a product of a human-built and curated rule set that can be published, and then contested, examined, and modified by others. This mitigates concerns over black box ML models. Novel ML or DL approaches can be severely limited by available datasets, with no widely accepted approach to create novel ones as needed and in an efficient

manner. CRAML-built models can have real-world applicability, because training data can be built and deployed by domain experts.

In response to the challenges introduced in the previous section, CRAML's methodological transparency is intended to address several of the problems with proprietary data and otherwise unreachable methods. A user can control what information is extracted from the corpus, and this therefore mitigates some concerns regarding sensitive data. With a standard desktop computer and open-source software, CRAML enables interactive engagement with massive text corpora without prior programming knowledge (although users will find some programming knowledge helpful and advanced users may seek to customize the underlying code for niche purposes). CRAML and other hybrid systems do have limitations: challenges include the ability to analyze the impact of iterative human interventions (Xin et al. 2018). However, at each crucial step of the framework, CRAML saves and shows the user the underlying data in CSV format to enable the interactivity essential to achieving a "human in the loop." Each step of the process is recorded for replication and transparency purposes. The artifacts created and used by CRAML in the intermediate stages enable users to see the effects of small changes to rules.

An Application of CRAML: Proprietary Job Advertisement Text

Search frictions harm both workers and employers by increasing the barriers to good matches in the labor market and reducing competition for labor (Burdett and Mortensen 1998; Manning 2005). A suite of ML classifiers that add niche filters to job search could benefit jobseekers, employers, and workforce agencies. With support from the National Science Foundation, the National Labor Exchange created the NLx Research Hub in 2021 to "increase the amount of actionable labor market information in the U.S. to facilitate the recruitment, hiring, and training opportunities of American workers" and "deepen partnerships between industry, government,

and academia by enhancing the infrastructure to support the convergence of research, education, and talent pipelines."

Workers who desire more flexible, remote jobs after COVID-19, disabled workers seeking specific working conditions, union workers, workers with a professional license, workers on visas, and veterans as well as veteran's spouses may all be underserved even while being preferred by specific employers. Job seekers may easily become discouraged without information on whether, for example: a job is located near public transit, or whether an employer will interview an ex-felon, a teen, or a non-resident on a visa. If some employers seek to target and hire underserved audiences, but job ads never reach the workers, then research should aim to reliably identify which jobs might be especially relevant for a particular worker.

Research application CRAML was first developed using job advertisement text data to support research on the labor market. While the underlying text is confidential and licensed from NLX, we describe and release rules files and nine ML classifiers under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. These classifiers achieve a high accuracy in detecting job characteristics and barriers to employment. Such application could improve both research and contribute to reducing barriers to employment in job advertisements. Ultimately, better data on employment practices obtained from job advertisement text and other sources of text information could have a profound impact on research capabilities in this field. The research implications of this work are not only research papers and data, but the publication of custom-built ML classifiers to create open tools for real-time information systems that track changes in the labor market. In Appendix B, we discuss how the broader labor market research community can contribute to this effort and an open-source ecosystem can work.

A STEP-BY-STEP, REPLICABLE NO POACH CLASSIFIER

We demonstrate the end-to-end CRAML approach by applying it to an empirical context where the resulting analysis might be meaningful, transparent, and replicable. Toward this end, we contribute a new text corpus of 151,708 franchise documents, and the cleaned and pre-processed text we used in this analysis. We publish metadata that tracks which records a particular document corresponds to, and includes the effective date, company name, and unique ID. We also publish the rule sets that are the manual hand-coded input, enabling replication and scrutiny of the results. Lastly, we publish the training dataset and the ML model.[2]

Empirical and Data Context

"No poach" clauses in contracts are anti-competitive restraints on employee mobility, and have drawn interest from academics, regulators, and policy-makers. While anti-competitive language is sometimes contrary to public policy, these documents are inaccessible for many: available only in massive, unsearchable collections of PDF documents on state agency websites. With subtle variations in language and no guide about where to look for these clauses, no poach clauses are diamonds in the rough: a few sentences in hundreds of thousands of documents with millions of pages. Indeed, such clauses were largely not known to exist by the relevant academic and policy community of interest until the release of a 2017 working paper (Krueger and Ashenfelter 2022). The paper contained a limited sample and relied on a third-party data provider to identify no poach clauses. Following the release of the working paper, eleven state attorneys general issued a letter in 2018 demanding the practice end, many franchises voluntarily

[2] The PDFs are hosted in partnership with DocumentCloud, and will be released upon publication. On DocumentCloud, they can now be searched, discussed, and annotated. The replication materials including extracted text, rules files, and machine learning classifier are available at 10.5281/zenodo.7454758.

ended the practice, and the State of Washington in 2018 first negotiated settlements with franchise companies in which many removed their no poach clauses.[3]

No poach clauses place downward pressure on wages (Callaci et al. 2022; Balasubramanian et al. 2022). Given their importance in reducing opportunities for job mobility, descriptive statistics regarding the prevalence of these clauses are of great interest. Overcoming the inaccessibility of the text within public documents is a significant challenge – a "herculean task" (Larsen and Bong 2016). We assembled a large collection of Franchise Disclosure Documents (FDDs) from the state of California by scraping PDF documents from the public web. These records exclude companies that may be exempt from filing and may be missing observations.[4]

In addition, we make available the machine-readable text corpus we build from the downloaded PDFs and the metadata that ties documents to specific company filings. For pre-processing PDFs to text, we used Python's Tika, and for files that could not be initially read with this, we used ABBYY Fine Reader. For the California corpus, we assembled 151,708 documents in 6.99 GB of cleaned machine-readable text that can be traced back to a total of 12,992 franchise records. We consider the record to be the unique ID that the state assigns to multiple documents filed by a franchise on a particular date. We trace each document back to a record with metadata that includes the name of the franchise, and the date of the filing.

[3] See Washington State Attorney General Report, Letter from State Attorneys General. As Ashenfelter wrote, "it is instructive that the mere revelation of collusive agreements, whether legal or not, has so quickly provoked a strong response from both the antitrust authorities and the franchisors whose agreements contained these no-poach clauses" (qtd. in Krueger and Ashenfelter (2022))

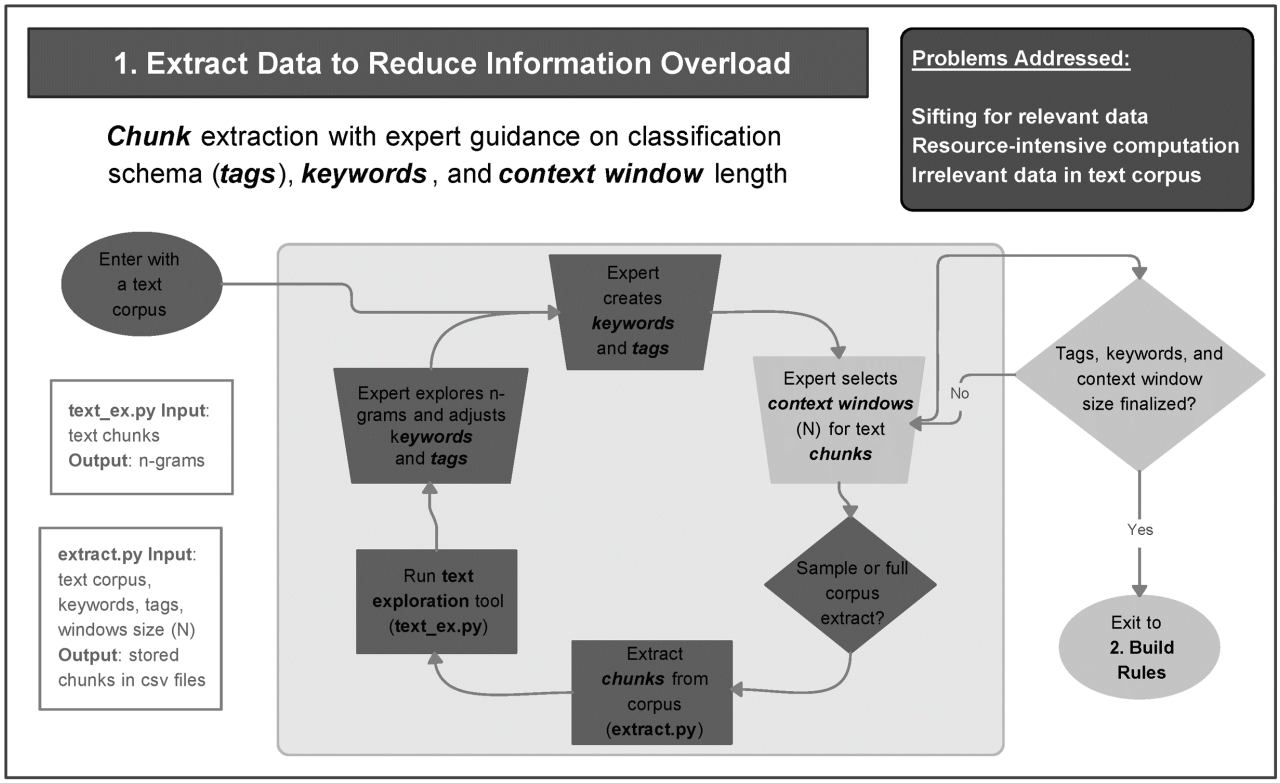
[4] We were able to obtain and match the 151,708 documents to 12,992 records via web scraping from the California DFPI portal, where we identified 16,216 franchise disclosure record IDs from January 2013-July 2022. We are investigating additional methods to acquire any missing data, which we believe to be randomly distributed.

The Construction of Training Data

Step 1: Extract Relevant Text to Reduce Information Overload

The extraction process encompasses taking the unstructured text files and preparing a cleaned dataset of only the text that is plausibly relevant and ready for hand-coding. By reducing the size of the text files being worked with, and delimiting the extracts to the relevant text in the corpus, this step addresses challenges of computational resource intensity, and assists in sifting for relevant information in a large corpus. This is a manual, iterative process – a loop that is exited when the user is satisfied that all of the relevant text is extracted from the corpus. Figure 3 illustrates the extraction process, which corresponds to Step 1 outlined in Figure 2. Two main software tools (extract.py and text_ex.py) aid the user in finding and exploring the relevant text.

Figure 3: First process in the CRAML Framework – Extract Data



We chose initial keywords “hire”, “recruit”, “employ”, and “solicit” to explore if franchise documents contain a no poach clause. The initial keywords chosen suggested additional ideas for tags, which in turn, suggested new ideas for keywords. To aid in this process, a repeated n-gram exploration helped discover the contexts in which given keywords appear the most. This exploration prioritizes the extract, and it suggests new ideas for keywords, tags, rules, and context windows. Because the full context was important to explore initially, the earliest exploratory extracts focused on the 2 sentences surrounding each keyword. Following all of the processes described below, a final context window of 13 words was selected (6 words to the left and right of the keyword). The goal when removing irrelevant or non-essential information is to reduce noise and build a highly focused ML classification model. In an iterative and flexible process, more keywords were subsequently added: for example, “poach”, “non compet”, “noncompet”, “covenant” and “not to compete” simply restarted the processes of Step 1.

Step 2: From Rule Sets to (Training) Data via Extrapolation, Testing, and Validation

After extraction, an expert conceptualizes and defines the desired classifications, or *tags*, and builds a rule set for each tag in a manual and iterative fashion. Figure 4 details the iterative process of Step 2 that involves validation, extrapolation, and the refinement of rule sets. This process concludes when the expert is satisfied that the extrapolated rules are valid on a random sample of the text. The rule set is then “extrapolated” to the entirety of the corpus, yielding a structured dataset of accurately classified chunks – a training dataset, in other words. The results of Step 2 can be used immediately for descriptive research regarding the contents of documents, or as a basis for training ML models.

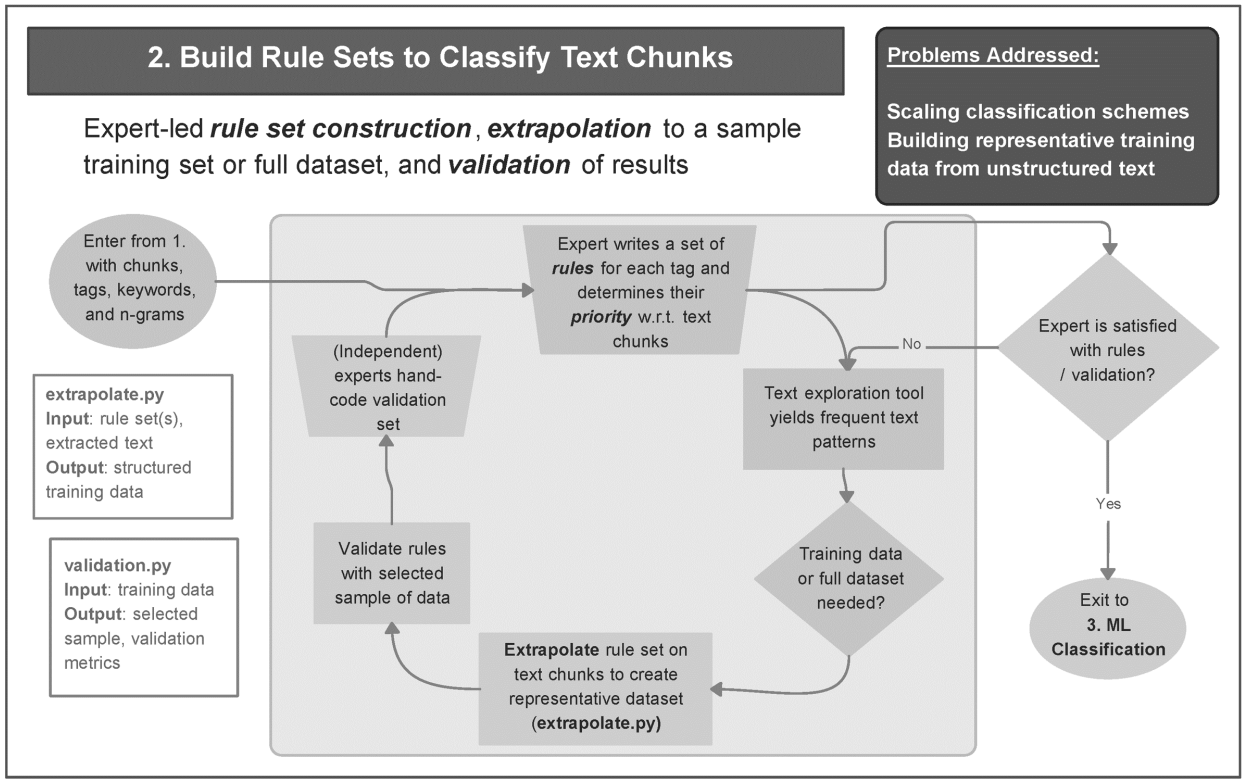
The cycle in Step 2 begins with the initial creation of rule sets for each tag, which can be done manually by an individual researcher, or team of researchers. Constructing and designing

accurate indicators requires researcher familiarity with ontological subtleties and appropriate construct validation and scale development methodology (Weber 2021; MacKenzie, Podsakoff, and Podsakoff 2011). Rules are written in a CSV file that contains a list of rules, the priority level of each rule, and the corresponding tag, assigned a 0 or 1. As earlier, the n-gram tool may be helpful – doing this prioritizes the creation of rules that will "cover" the largest parts of the entire data. Priority level determines the order in which a rule is run. Earlier rules can be over-written by a subsequent priority rule.

Step 2 then involves the use of rule sets and the extrapolation algorithm (extrapolate.py) to validate the rules. In essence, the extrapolation process converts the set of rules into an extended dataset, and tags each chunk according to the user's encoding of each rule. The resulting dataset contains the unique document identifier, the extracted chunk matching a certain rule, and the defined encoding for this chunk. Note that this extrapolation is done individually for each rule set and tag, thus resulting in a training dataset for each rule set (file). A second software tool (validate.py) relates to the validation of the data that is the output of extrapolation. Once extrapolated, the expert tests the performance of the rules against actual chunks that are coded by those rules. The expert analyses a strategic sample of chunks meant to ensure each rule is valid – choosing a random sample of N examples per rule. This enables independent hand-coders to score each chunk, which can then be re-imported to assess accuracy and performance of rules.

For the franchise document data, allowing iterative exploration of keywords and adjustments to rules, we ultimately developed a set of 324 rules to classify suspect no poach clauses in the corpus. As one rule for the no poach tag states: "you may not seek to employ or retain any employee or independent contractor who is at any time employed by us."

Figure 4: Second process in the CRAML Framework – Build Rules



CRAML emphasizes content validation processes at the input stage to ML. Independent validation in which a third researcher hand-coded 706 chunks indicates an initial 91% match between an independent third party and the extrapolation of rules, suggesting a high degree of inter-rater reliability in ability to detect characteristics of no poach clauses. While our emphasis is to ensure that there is strong inter-rater reliability between the CRAML user and an independent observer in this test example, we do not claim perfect identification of all no poach clauses or any legal conclusions. For a regulator or lawyer, additional scrutiny would be required; our goal is to identify and classify with precision the no poach language within an ocean of text and pinpoint its location. Step 2 yields a dataset for training a ML classification model, or, if the underlying corpus is small enough (as is the case here), rules can be applied to the entire dataset and reveal which filings contain tags according to the specified rules.

Step 3: Training ML Classifiers to Build Structured Datasets

While an extrapolation of rules to all chunks in the dataset is possible, CRAML also yields data on which a ML classification model, or classifier, can be trained. ML has the advantage over exact rules of recognizing new and unseen instances, as well as overcoming messy data and inconsistent use of language in unique situations. Although many algorithms exist for such a training process, the CRAML framework trains a binary classifier for each tag. In other words, each training dataset is created from a rules file, which contains one or more tags. Accordingly, each classifier can be traced back to a rule set to perform a 0/1 classification for the tags included therein. The training process is described in greater detail in Appendix C. Once the classifier is trained, it can be deployed to classify the original data, i.e., the entire unstructured text corpus, or can be used on other text sources. This represents the completion of the CRAML framework, as one can now build a structured, labeled dataset from the unstructured text of a document corpus using a ML model. This entire final process is illustrated in Figure 5.

Evaluating the Model

In this empirical context, we train a no poach classifier built on training data from the franchise disclosure documents. We use the classifier to build datasets and compare the rules-based approach and the ML approach using the same data. The "no poach" classifier is built with training data from a 10% sample plus all positive instances (no poach tag=1) in the full sample. This is done to combat class imbalance, as positive instances are very rare. Thus, when the occurrence of positive or negative tags is very low, manual augmentations to training data such as this are required to receive acceptable model performance. We obtain an accuracy score of 0.99, precision of 0.97, recall of 0.96, and an overall F1-Score of 0.97. The harmonic mean of

precision and recall (or F1-score) is based upon a comparison between the ML output and the training dataset.

Figure 5: Third process in the CRAML Framework – Train Classifiers

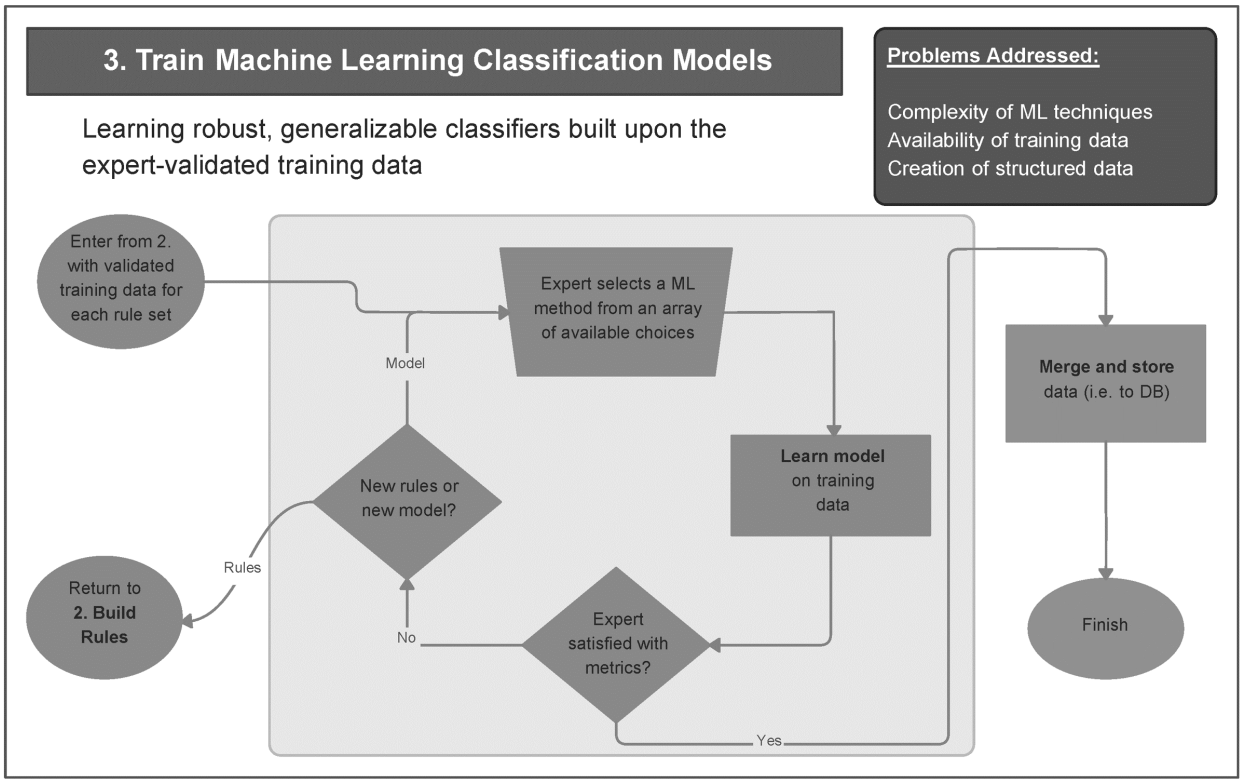
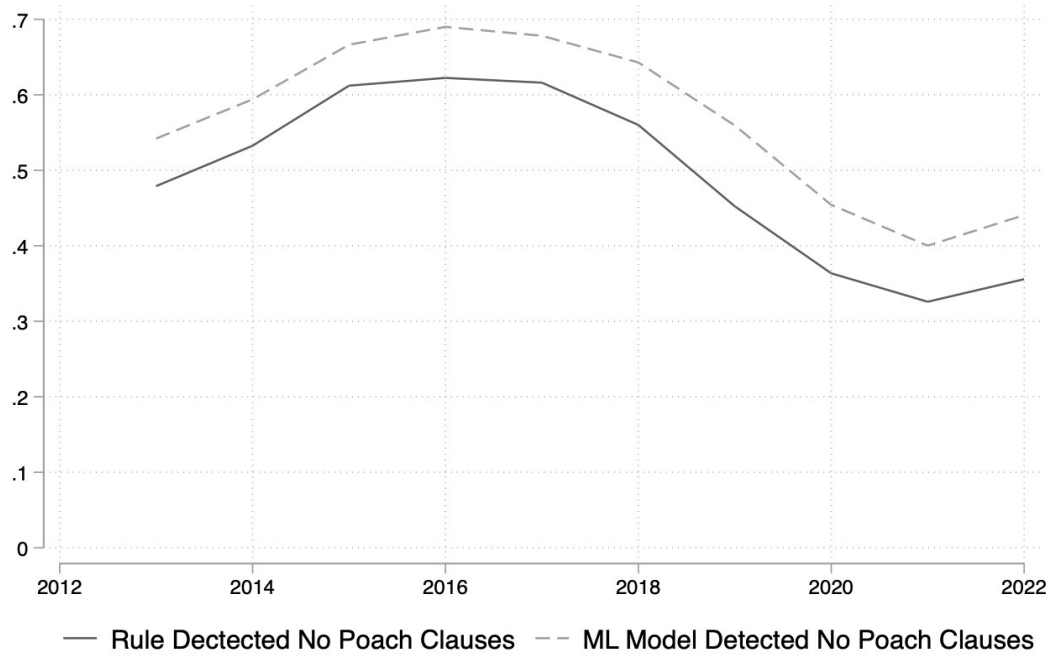


Figure 6 displays the percent of all rule-detected and ML-detected no poach clauses in the California corpus within each year for each record filed from years 2013-2022 (with partial data for 2022). As seen in Figure 6, the ML classifier identifies more suspected no poach clauses than the rules, and this increases in 2021. Additional research using this corpus is pinpointing fine-grained characteristics of language in these documents that restrict employee mobility. Even so, the ML-detected pattern is similar to the rule-based analysis. It shows that from 2015-2017, over 60% of the records contain suspect no poach clauses, after which there was a decline until the percent stabilizes below 40%. This suggests that the interventions that followed Krueger and Ashenfelter (2022) decreased the prevalence of no poach language in franchise documents. The

continued prevalence of these clauses, despite interventions from authorities, requires further investigation: while no poach clauses continue to exist, inspection of the text chunks reveals that while some remain the same as before the intervention of the Washington State Attorney General, others now limit their applicability to more favorable jurisdictions or restrict their application to highly compensated employees.

Figure 6: Analysis of Suspect No Poach Clauses in California Franchise Documents



The above statistics reflect performance of the model at the "chunk" level. If an expert were looking to confirm that certain documents contain no poach clauses, the record is the level of analysis where they would want to begin their search. Given over 150,000 documents stored in nearly 13,000 records, this could be a daunting task. Moving to the level of the records, the no poach classifier closely matches the rule-generated results, with only 22 false negatives and 672 false positives in 12,922 records. Thus, at the record level, ML model achieves a 0.95 F1-score, 0.996 recall, 0.91 precision, and accuracy of 0.946.

A further inspection of chunks reveals that a significant portion of the false positives are in fact related to other language that accompanies no poach clauses and belong properly in the context of anti-competitive language in these documents: jurisdictional restrictions on no poach clauses, language that applies no poach clauses only to specific employees, and language that creates non-compete clauses. Precision at the level of the record would be 0.04 higher (0.95) if one concludes these clauses are relevant to a search for anti-competitive language.

The results of CRAML are fully traceable to choices made in the construction of a rule set for the no poach classifier. If one were dissatisfied with the model performance, or wanted to adapt the specific definition of the no poach construct, it would be possible to "steer learning" and achieve a desired result by changing rule sets and thus re-shaping the training data. A user could add or remove all jurisdictional, narrow, and non-compete language from the no poach rule set to achieve a more discriminating classifier. A user could also combine all the rules to make a single classifier that memorizes a larger construct related to all types of anti-competitive language that appear in the documents.

DISCUSSION

The CRAML method enables the systematic and structured process of creating novel datasets from unstructured text corpora. The result harmonizes advanced, automated information extraction and ML techniques with the input and expertise from manual analysis performed by supervising researchers. The exploration and iterative process performed during the context rule creation stage involves work that is not only difficult to perform automatically, but also that will arguably never come close to matching the diversity and expertise of human experts with domain knowledge. As a result, this intermediate stage in the framework incorporates a human element lacking in modern classification frameworks.

Future Directions

Technical Improvements

Future research directions regarding the technical backbone of the CRAML framework aim to bolster the extrapolation stage of the pipeline. Embeddings can improve the transformation of context rules to datasets (Li, Larsen, and Abbasi 2020). With embeddings added to the extrapolation method, the need for training a ML classifier in the end may become obsolete. A second area of future work involves building tools to aid the domain expert in the initial, manual-driven stages of the CRAML process. Keyword extraction, topic modeling, automatic rule induction, and knowledge graphs could all be useful tools in supporting the domain expert to express his or her worldview, as well as explore large text corpora in a richer way. Text generation models could also assist in augmenting and building custom training data.

Although the focus is currently placed on ML techniques, the role of more advanced classifiers remains an open area of investigation. Deep Learning, in particular sequential models, could prove to be powerful in boosting the predictive capability of models trained on CRAML-generated datasets. Of course, this would come with the added overhead of extra training and parameter tuning, which must also be considered for future endeavors.

Computational Social Science

CRAML is designed with flexibility and web-based sources of text in mind. A plug-in was created to integrate DocumentCloud (used by journalists and academics to post government documents) for the project related to no poach agreements. New plug-ins could draw data from web-based sources of text such as Twitter. To address a concern in Porway (2022), government, civil society, and non-governmental organizations could develop context rule sets to capture text relevant to their interests and publish or license ML classifiers that provide useful feeds and

classifications of information tracking on topics of societal concern. For academics, editors and reviewers could demand greater transparency when claims are made about data built from text.

Finally, experts in many fields can hopefully develop diverse and practical applications that uncover constructs and patterns currently "hidden" in unstructured text. The CRAML framework and software tools can be used to analyze text in any language. To the extent a construct is intra-subjectively reliable, it could be developed by a researcher into rules and scaled over vast amounts of unstructured text. To the extent that inter-subjective variance is high, we hope that by decreasing the time and effort needed to scale a framework, there will be many opportunities for diverse voices to produce contesting interpretative schemes that are transparent and replicable.

References

Abbasi, A., Zhang, Z., Zimbra, D., & Chen, H. (2010). Detecting Fake Websites: The Contribution of Statistical Learning Theory. *MIS Quarterly*, *34*(No. 3), 435–461. https://doi.org/10.2307/25750686

Abbas, A., Zhou, Y., Deng, S., & Zhang, P. (2018). Text Analytics to Support Sense-Making in Social Media: A language-Action Perspective. *MIS Quarterly*, *42*(2), 427–464. https://doi.org/10.25300/MISQ/2018/13239

Abbasi, A., Albrecht, C., Vance, A., & Hansen, J. (2012). Metafraud: A Meta-Learning Framework for Detecting Financial Fraud. *MIS Quarterly*, *36*(4), 1293–1327. https://doi.org/10.2307/41703508

Abbott, A. (2001). *Chaos of Disciplines*. The University of Chicago Press.

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

Adnan, K., & Akbar, R. (2019). Limitations of information extraction methods and techniques for heterogeneous unstructured big data. *International Journal of Engineering Business Management*, *11*. https://doi.org/10.1177/1847979019890771

Aggarwal, C. C., & Zhai, C. (2012). A Survey of Text Clustering Algorithms. In *Mining text data* (pp. 77–128). Springer.

Apprich, C. (2018). Secret Agents: A Psychoanalytic Critique of Artificial Intelligence and Machine Learning. *Digital Culture & Society*, *4*(1), 29–44. https://doi.org/10.14361/dcs-2018-0104

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2019). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *ArXiv:1910.10045 [Cs.AI]*, *2*, 1–72. https://doi.org/10.48550/arXiv.1910.10045

Asatiani, A., Malo, P., Nagbøl, P. R., Penttinen, E., Rinta-Kahlia, T., & Salovaara, A. (2020). Challenges of Explaining the Behavior of Black-Box AI Systems. *MIS Quarterly*, *19*(4), 259-278.

Babic, B., Cohen, I. G., Evgeniou, T., & Gerke, S. (2021, January 1). When Machine Learning Goes Off the Rails. *Harvard Business Review*.

Bachura, E., Valecha, R., Chen, R., & Rao, H. R. (2022). The OPM Data Breach: An Investigation of Shared Emotional Reactions on Twitter. *MIS Quarterly*, *46*(2), 881–910.

Balasubramanian, N., Chang, J. W., Sakakibara, M., Sivadasan, J., & Starr, E. (2022). Locked In? The Enforceability of Covenants Not to Compete and the Careers of High-Tech Workers. *Journal of Human Resources*, *57*(S), S349–S396. https://doi.org/10.3368/jhr.monopsony.1218-9931R1

Bapna, S., Benner, M. J., & Qiu, L. (2019). Nurturing Online Communities: An Empirical Investigation. *MIS Quarterly*, *43*(2), 425–452. https://doi.org/10.25300/MISQ/2019/14530

Begum, S. H., & Nausheen, F. (2018). A comparative analysis of differential privacy vs other privacy mechanisms for Big Data. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, 512–516. https://doi.org/10.1109/ICISC.2018.8399125

Benbya, H., Davenport, Thomas. H., & Pachidi, S. (2020). Artificial intelligence in organizations: Current state and future opportunities. *MIS Quarterly Executive*, *19*(4), 1–15.

Bradley, V. C., Kuriwaki, S., Isakov, M., Sejdinovic, D., Meng, X.-L., & Flaxman, S. (2021). Unrepresentative big surveys significantly overestimated US vaccine uptake. *Nature*, *600*(7890), Article 7890. https://doi.org/10.1038/s41586-021-04198-4

Broek, E. V. D., Sergeeva, A., & Huysman, M. (2021). When the Machine Meets the Expert: An Ethnography of Developing AI for Hiring. *MIS Quarterly*, *45*(3), 1557–1580. https://doi.org/10.25300/MISQ/2021/16559

Burdett, K., & Mortensen, D. T. (1998). Wage Differentials, Employer Size, and Unemployment. *International Economic Review*, *39*(2), 257–273. https://doi.org/10.2307/2527292

Burkart, N., & Huber, M. F. (2021). A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, *70*, 245–317. https://doi.org/10.1613/jair.1.12228

Callaci, B., Pinto, S., Steinbaum, M., & Walsh, M. (2022). *The Effect of No-poaching Restrictions on Worker Earnings in Franchised Industries* (SSRN Scholarly Paper No. 4155577). https://doi.org/10.2139/ssrn.4155577

Card, D., Chetty, R., Feldstein, M. S., & Saez, E. (2010). *Expanding Access to Administrative Data for Research in the United States* (SSRN Scholarly Paper No. 1888586). https://doi.org/10.2139/ssrn.1888586

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, *538*(7623), 20–23. https://doi.org/10.1038/538020a

Chau, M., Li, Tim. M. H., Wong, P. W. C., Xu, J. J., Yip, Paul. S. F., & Chen, H. (2020). Finding People with Emotional Distress in Online Social Media: A Design Combining Machine Learning and Rule-Based Classification. *MIS Quarterly*, *44*(2), 933–955. https://doi.org/10.25300/MISQ/2020/14110

Choi, J., Menon, A., & Tabakovic, H. (2021). Using machine learning to revisit the diversification–performance relationship. *Strategic Management Journal*, *42*(9), 1632–1661. https://doi.org/10.1002/smj.3317

Choudhury, P., Allen, R. T., & Endres, M. G. (2020). Machine learning for pattern discovery in management research. *Strategic Management Journal*, *42*(1), 30–57. https://doi.org/10.1002/smj.3215

Cohen, L. E., & Mahabadi, S. (2022). In the Midst of Hiring: Pathways of Anticipated and Accidental Job Evolution During Hiring. *Organization Science*, *33*(5), 1938–1963. https://doi.org/10.1287/orsc.2021.1516

Cutler, A. D., Carden, S. W., Dorough, H. L., & Holtzman, N. S. (2021). Inferring Grandiose Narcissism From Text: LIWC Versus Machine Learning. *Journal of Language and Social Psychology*, *40*(2), 260–276. https://doi.org/10.1177/0261927X20936309

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C., Mincu, D., … Sculley, D. (2022). Underspecification Presents Challenges for Credibility in Modern Machine Learning. *Journal of Machine Learning Research, 23*.

Dautenhahn, K. (1998). The Art of Designing Socially Intelligent Agents: Science, Fiction, and the Human in the Loop. *Applied Artificial Intelligence*, *12*(7–8), 573–617. https://doi.org/10.1080/088395198117550

Davis, G. F. (2015). Editorial essay: What is organizational research for? *Administrative Science Quarterly*, *60*(2), 179–188.

DiMaggio, P. (2015). Adapting computational text analysis to social science (and vice versa). *Big Data & Society*, *2*(2), 2053951715602908. https://doi.org/10.1177/2053951715602908

Duriau, V. J., Reger, R. K., & Pfarrer, M. D. (2007). A Content Analysis of the Content Analysis Literature in Organization Studies: Research Themes, Data Sources, and Methodological Refinements. *Organizational Research Methods*, *10*(1), 5–34. https://doi.org/10.1177/1094428106289252

Faik, I., Barrett, M., & Oborn, E. (2020). How Information Technology Matters in Societal Change: An Affordance-Based Institutional Perspective. *MIS Quarterly*, *44*(3), 1359–1390. https://doi.org/10.25300/MISQ/2020/14193

Geva, H., Saar-Tsechansky, M., & Oestreicher-Singer, G. (2019). Using retweets when shaping our online persona: Topic modeling approach. *MIS Quarterly*, *43*(2), A1–A20.

Gillespie, T. (2020). Content moderation, AI, and the question of scale. *Big Data & Society*, *7*(2), 2053951720943234. https://doi.org/10.1177/2053951720943234

Giorgi, S., Maoret, M., & J. Zajac, E. (2019). On the Relationship Between Firms and Their Legal Environment: The Role of Cultural Consonance. *Organization Science*, *30*(4), 803–830. https://doi.org/10.1287/orsc.2018.1250

Glaser, B. G., & Strauss, A. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction.

Gorwa, R., Binns, R., & Katzenbach, C. (2020). Algorithmic content moderation: Technical and political challenges in the automation of platform governance. *Big Data & Society*, *7*(1), 2053951719897945. https://doi.org/10.1177/2053951719897945

Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., & Yang, G.-Z. (2019). XAI—Explainable artificial intelligence. *Science Robotics*, *4*(37).

Guo, W., Sengul, M., & Yu, T. (2021). The Impact of Executive Verbal Communication on the Convergence of Investors' Opinions. *Academy of Management Journal*, *64*(6), 1763–1792. https://doi.org/10.5465/amj.2019.0711

Hammoud, K., Benbernou, S., Ouziri, M., Saygın, Y., Haque, R., & Taher, Y. (2019). Personal information privacy: What's next? *CEUR Workshop Proceedings*.

Hargittai, E. (2015). Is Bigger Always Better? Potential Biases of Big Data Derived from Social Network Sites. *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 63–76. https://doi.org/10.1177/0002716215570866

Heaven, W., Douglas. (2020, November 18). *The way we train AI is fundamentally flawed*. MIT Technology Review.

Hedderich, M. A., Lange, L., Adel, H., Strötgen, J., & Klakow, D. (2021). *A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios* (arXiv:2010.12309 (cs); pp. 1–24). arXiv. https://doi.org/10.48550/arXiv.2010.12309

Kane, G. C., Young, Amber. G., Majchrzak, A., & Ransbotham, S. (2020). Avoiding an oppressive future of machine learning: A design theory for emancipatory assistants. *MIS Quarterly*, 45(1), 371-396.

Kaplan, S., & Vakili, K. (2014). The double-edged sword of recombination in breakthrough innovation. *Strategic Management Journal*, *36*(10), 1435–1457. https://doi.org/10.1002/smj.2294

Kapoor, S., & Narayanan, A. (2022). *Leakage and the Reproducibility Crisis in ML-based Science* (arXiv:2207.07048). arXiv. https://doi.org/10.48550/arXiv.2207.07048

King, G. (2011). Ensuring the Data-Rich Future of the Social Sciences. *Science*, *331*(6018), 719–721. https://doi.org/10.1126/science.1197872

Krueger, A. B., & Ashenfelter, O. (2022). Theory and Evidence on Employer Collusion in the Franchise Sector. *Journal of Human Resources*, *57*(S), S324–S348. https://doi.org/10.3368/jhr.monopsony.1019-10483

Lacity, M., & Willcocks, L. (2021). Becoming strategic with intelligent automation. *MIS Quarterly Executive*, *20*(2), 1–14. https://doi.org/10.17705/2msqe.000XX

Larsen, Kai. R., & Bong, Chih. How. (2016). A Tool for Addressing Construct Identity in Literature Reviews and Meta-Analyses. *MIS Quarterly*, *40*(3), 529–551.

Lazer, D. M. J., Pentland, A., Watts, D. J., Aral, S., Contractor, N., Freelon, D., Gonzalez-Bailon, S., King, G., Margetts, H., Nelson, A., Salganik, M. J., Strohmaier, M., Vespignani, A., & Wagner, C. (2020, August 28). Computational social science: Obstacles and opportunities. *Science, 369*(6507) 1060-1062. https://doi.org/10.1126/science.aaz8170

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & Van Alstyne, M. (2009). Computational Social Science. *Science*, *323*(5915), 721–723. https://doi.org/10.1126/science.1167742

Lazer, D., & Radford, J. (2017). Data ex Machina: Introduction to Big Data. *Annual Review of Sociology*, *43*(1), 19–39. https://doi.org/10.1146/annurev-soc-060116-053457

Li, H., Zhang, C., & Kettinger, William. J. (2022). Digital Platform Ecosystem Dynamics: The Roles of Product Scope, Innovation, and Collborative Network Centrality. *MIS Quarterly*, *46*(2), 739–770. https://doi.org/10.25300/MISQ/2022/15444

Li, J., Larsen, K., & Abbasi, A. (2020). TheoryOn: A design framework and system for unlocking behavioral knowledge through ontology learning. *MIS Quarterly*, *44*(4), 1733–1772. https://doi.org/10.25300/MISQ/2020/15323

Liu, Angela. X., Li, Yillin., & Xu, Sean. Xin. (2021). Assessing the Unacquainted: Inferred Reviewer Personality and Review Helpfulness. *MIS Quarterly*, *45*(3), 1113–1148.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. https://aclanthology.org/P11-1015

MacKenzie, Scott. B., Podsakoff, Philip. M., & Podsakoff, Nathan. P. (2011). Construct Measurement and Validation Procedures in MIS and Behavioral Research: Integrating New and Existing Techniques. *MIS Quarterly*, *35*(2), 293–334. https://doi.org/10.2307/23044045

Manning, A. (2005). *Monopsony in Motion: Imperfect Competition in Labor Markets*. Princeton University Press. https://press.princeton.edu/books/paperback/9780691123288/monopsony-in-motion

McKenny, A. F., Short, J. C., & Payne, G. T. (2013). Using Computer-Aided Text Analysis to Elevate Constructs: An Illustration Using Psychological Capital. *Organizational Research Methods*, *16*(1), 152–184. https://doi.org/10.1177/1094428112459910

Miranda, Shaila. M., Wang, Dawei. D., & Tian, Chuan. A. (2022). Discursive Fields and the Diversity-Coherence Paradox: An Ecological Perspective on the Blockchain Community Discourse. *MIS Quarterly*, *46*(3), 1421–1452. https://doi.org/10.25300/MISQ/2022/15736

Munger, K. (2019). The Limited Value of Non-Replicable Field Experiments in Contexts With Low Temporal Validity. *Social Media + Society*, *5*(3), 2056305119859294. https://doi.org/10.1177/2056305119859294

Munger, K., Guess, A. M., & Hargittai, E. (2021). Quantitative Description of Digital Media: A Modest Proposal to Disrupt Academic Publishing. *Journal of Quantitative Description: Digital Media*, *1*, 1–13. https://doi.org/10.51685/jqd.2021.000

Nelson, L. K. (2017). Computational Grounded Theory: A Methodological Framework. *Sociological Methods & Research*, *49*(1), 3–42. https://doi.org/10.1177/0049124117729703

Nelson, L. K. (2019). To Measure Meaning in Big Data, Don't Give Me a Map, Give Me Transparency and Reproducibility. *Sociological Methodology*, *49*(1), 139–143. https://doi.org/10.1177/0081175019863783

Pandey, S., & Pandey, S. K. (2017). Applying Natural Language Processing Capabilities in Computerized Textual Analysis to Measure Organizational Culture. *Organizational Research Methods*, *22*(3), 765–797. https://doi.org/10.1177/1094428117745648

Pfeffer, J., Mooseder, A., Hammer, L., Stritzel, O., & Garcia, D. (2022). This Sample seems to be good enough! Assessing Coverage and Temporal Reliability of Twitter's Academic API. *ArXiv:2204.02290 [Cs]*, *2*, 1–10.

Porway, J. (2022). Funding Data and AI That Serve the Social Sector. *Stanford Social Innovation Review*. https://doi.org/10.48558/YKY9-RX35

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206-215.

Short, J. C., Broberg, J. C., Cogliser, C. C., & Brigham, K. H. (2010). Construct validation using computer-aided text analysis (CATA): An illustration using entrepreneurial orientation. *Organizational Research Methods*, *13*(2), 320–347. https://doi.org/10.1177/1094428109335949

Short, J. C., & Palmer, T. B. (2008). The Application of DICTION to Content Analysis Research in Strategic Management. *Organizational Research Methods*, *11*(4), 727–752. https://doi.org/10.1177/1094428107304534

Shrestha, Y. R., He, V. F., Puranam, P., & Von Krogh, G. (2021). Algorithm Supported Induction for Building Theory: How Can We Use Prediction Models to Theorize? *Organization Science*, *32*(3), 856–880. https://doi.org/10.1287/orsc.2020.1382

Sine, W. D., Cordero, A. M., & Coles, R. S. (2022). Entrepreneurship Through a Unified Sociological Neoinstitutional Lens. *Organization Science*, *33*(4), 1675–1699. https://doi.org/10.1287/orsc.2022.1586

Subramani, Mani., Wagle, Mihir., Ray, Gautam., & Gupta, Alok. (2021). Capability Development through Just-in-Time Access to Knowledge in Document Repositories: A Longitudinal Examination of Technical Problem Solving. *MIS Quarterly*, *45*(3), 1287–1308. https://doi.org/10.25300/MISQ/2021/15635

Tchalian, H. (2019). Microfoundations and Recursive Analysis: A Mixed-Methods Framework for Language-Based Research, Computational Methods, and Theory Development. In P. Haack, J. Sieweke, & L. Wessel (Eds.), *Microfoundations of Institutions* (Vol. 65B, pp. 107–125). Emerald Publishing Limited. https://doi.org/10.1108/S0733-558X2019000065B008

Tong, S., Jia, N., Luo, X., & Fang, Z. (2021). The Janus face of artificial intelligence feedback: Deployment versus disclosure effects on employee performance. *Strategic Management Journal*, *42*(9), 1600–1631. https://doi.org/10.1002/smj.3322

Wagstaff, K. L. (2012). Machine learning that matters. In *Proceedings of the 29th International Conference on International Conference on Machine Learning* (pp. 1851-1856).

Watts, D. J. (2013). Computational Social Science: Exciting Progress and Future Directions. *The Bridge on Frontiers of Engineering*, *43*(4), 7–10.
Weber, K. (2005). A toolkit for analyzing corporate cultural toolkits. *Poetics*, *33*(3), 227–252. https://doi.org/10.1016/j.poetic.2005.09.011
Weber, R. (2021). Constructs and Indicators: An Ontological Analysis. *MIS Quarterly*, *45*(4), 1644–1678. https://doi.org/10.25300/MISQ/2021/15999
Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., & He, L. (2022). A Survey of Human-in-the-loop for Machine Learning. *Future Generation Computer Systems*, *135*, 364–381. https://doi.org/10.1016/j.future.2022.05.014
Xin, D., Ma, L., Liu, J., Macke, S., Song, S., & Parameswaran, A. (2018). Accelerating Human-in-the-loop Machine Learning: Challenges and Opportunities. *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, 1–4. https://doi.org/10.1145/3209889.3209897
Yue, W. T., Wang, Q. H., & Hui, K. L. (2019). See no evil, hear no evil? Dissecting the impact of online hacker forums. *MIS Quarterly*, *43*(1), 63–95. https://doi.org/10.25300/MISQ/2019/13042
Zanzotto, F. M. (2019). Viewpoint: Human-in-the-loop Artificial Intelligence. *Journal of Artificial Intelligence Research*, *64*, 243–252. https://doi.org/10.1613/jair.1.11345
Zhang, Z., Hummel, J. T., Nandhakumar, J., & Waardenburg, L. (2020). Addressing the Key Challenges of Developing Machine Learning AI Systems for Knowledge-Intensive Work. *MIS Quarterly Executive*, *19*(4), 221–238. https://doi.org/10.17705/2msqe.00035

APPENDIX A: USING CRAML FOR A LITERATURE REVIEW

Literature search often begins in an academic search engines that have difficulty retrieving information from deep within a paper. CRAML allowed us to mine text from methods and data sections of publications, identify plausibly relevant papers, and read and incorporate into the literature review. We assembled the corpus and PDFs in Zotero, exported the metadata, converted all PDFs to text, minimally cleaned the text, and used Steps 1-3 of the CRAML process and software to identify relevant papers with keywords, tags, and rules. We deductively generated tags and keywords, shown in Listing A.1. The *keywords.json* file is used to extract all keyword-containing chunks of text. After retrieving 13-word chunks, we studied n-grams and developed rules files for each tag to classify the chunks that were relevant. Keywords alone generated numerous false positives. Machine Learning was not desirable or attempted. This effort could be further refined, but we stopped when the results produced the relevant papers.

Listing A.1. Management Corpus Tags and Keywords (*keywords.json*)

```
{
 "is_text": ["text", "corpus", "document", "repository", "mining" ],
"CATA": ["computer aided text analysis", "cata", "computational text", "computationally
      aided", "dictionary", "word count"],
"ML": ["machine learning", "deep learning", "supervised learning"],
"AI": ["artificial intelligen", "artificially intelligen", "neural network"],
"NLP": ["nlp", "natural language process", "embedding"],
"topic_model": ["lda", "latent dirichlet allocation", "topic model", "unsupervised learning", "tf
      idf", "tfidf", "inverse document frequency"],
"proprietary": ["license", "proprietary", "confidential", "secret", "sensitive"]
}
```

Rules that define the tag *istext* or "is text" identify papers that refer to a text corpus or text data. We also construct simple rule sets to identify papers that discuss *ML, NLP*, *Topic Model*, and *AI*. To make Figure 1, we focus only on observations where *istext* and another tag are both equal to 1, Literature search in CRAML is aided by use of regular expressions: keywords for proprietary data and models extracted many irrelevant chunks. To limit the focus to non-

transparency in data and models, we combine our keywords with the added terms "data" or "model" (e.g., "REGEX:::(? = .∗\bdata\b)((? = .∗\bconfidential\b)|(? = .∗\blicense)|(? = .∗\bsensitive)|(? = .∗\bproprietary\b)|(? = .∗\bsecret)).∗)".

CRAML's extrapolate function yields a CSV file merged with the metadata and tags constructed in CRAML. For each paper, CRAML provides a 0/1 indicator of whether or not the paper contains a chunk corresponding to the rules files above. Table A.1 aggregates the result of the CRAML analysis by year and theme.

Table A.1: Total Papers by Year that Reflect Rule Sets

Year	Proprietary Data or Models	Topic Models or LDA	ML	NLP	AI	CATA	Text or Document Corpus
2015	19	2	0	1	0	53	7
2016	11	0	1	0	0	68	10
2017	18	0	0	0	2	69	9
2018	20	2	0	2	2	73	17
2019	13	3	2	1	3	82	18
2020	17	2	2	1	1	67	18
2021	19	6	7	3	2	80	31

Because we are interested in papers that use text corpora, we filtered the metadata for only the papers that equal 1 for *istext*. The data for Figure 1 is presented in Table A.2.

Table A.2: Subset of Papers Where *istext*=1 by Year

Year	Proprietary Data or Models	Topic Models or LDA	ML	NLP	AI	CATA	Text or Document Corpus
2015	1	2	0	1	0	6	7
2016	2	0	1	0	0	8	10
2017	0	0	0	0	0	6	9
2018	1	1	0	1	1	11	17
2019	3	2	1	1	1	10	18
2020	2	1	2	0	1	13	18
2021	4	5	6	3	1	19	31

APPENDIX B: USING CRAML TO BUILD CLASSIFIERS FROM SENSITIVE DATA

Concerns with proprietary data sources and privacy can be partially addressed by the information extraction methods used in CRAML. While additional efforts are needed, including the potential development of further tools for incorporating Privacy-Enhancing Technologies such as Differential Privacy with text, the extraction process is a step toward efforts to protect individuals against identification and protect the data owner from violation of proprietary information and trade secrets. For example, researchers have been barred from accessing records due to privacy and confidentiality concerns, but with careful selection of keywords and removal of sensitive chunks through named entity recognition, experts could code chunks of extracted text and classify every document in a corpus without ever accessing the full or confidential information. Resulting classifiers and rules that create them can be made available to other academic researchers for transparency and replication while preserving underlying information. Once built, ML classifiers trained on de-identified or privatized chunks can be used to build datasets on any text corpus in the same domain.[5]

We release nine ML classifiers and their corresponding rules, created from a corpus of job advertisement text. Information found on https://www.petenorlander.com/research/work-with-me/ highlights how interested members of the research community can become involved.

References

Connolly, N. D. B., Winling, L., Nelson, R. K., & Marciano, R. (2018). Mapping inequality: 'Big data'meets social history in the story of redlining. In *The Routledge Companion to Spatial History* (pp. 502-524). Routledge.

[5] For example, researchers could track the frequency and spatial trends in 20th Century use of residential restrictive covenants (red-lining) that barred Black, Jewish, Asian and other ethnic groups from living in certain neighborhoods (Connolly et al. 2018). County commissioners of deeds have either barred researchers or severely limited access to digitized records due to concerns. For one project focused on Hennepin County, see https://mappingprejudice.umn.edu/.

APPENDIX C: THE CRAML PIPELINE

This appendix describes the CRAML methodology in detail. In Algorithm C.1, we extract context windows from a set of text documents, i.e. a corpus. This represents the important first step for handling large-scale unstructured text corpora by first extracting the "candidate diamonds" from the rough, to allow for the precise analysis of a domain expert.

Algorithm C.1: Extraction Pseudocode

```
Require: Text: collection of text documents, Keywords: list of defined keywords
Ensure: list of extracted files (location)
all_contexts = {}
for raw_text in documents do                    ▷ Note: done in parallel (unordered)
    contexts = []
    id = generate id
    if any keyword in text then
        cleaned = clean(text)                   ▷ text replace and regex cleaning
        for sentence in cleaned do
            if any keyword in sentence then
                c = get_context(sentence)       ▷ get context window
                contexts.append(c)
        contexts = set(contexts)
        all_contexts[id] = contexts
df = pd.DataFrame(all_contexts)
df.to_csv()
return list of saved filenames
```

As displayed in Algorithm C.1, tags and keywords are required for extraction. The extraction algorithm will extract any chunk of text where one of the keywords appears, and store the chunk and metadata about its location in the corpus inside a CSV file for human analysis. Users define a list of keywords for each tag, e.g.: *"nopoach"*: [*"solicit"*, *"employ"*, *"hire"*, *"recruit"*, *"covenant"*, *"staff"*, *"personnel"*].

Table C.1 illustrates the result of extraction: a CSV formatted dataset containing chunks of text, with a minimum of two fields: id and text (additional fields of document metadata can be included or excluded). The extraction algorithm retrieves chunks where keywords are exactly matched to a string of text. For example, the keyword "employ" will extract all variants that

contain that exact match (employment, employer, employee, unemployment, etc.). The ID matches the identification number from the metadata, and text represents a delimited list of context windows around extracted keywords. In this case, a context window with a defined parameter *n* refers to an extracted chunk of text with a maximum of *n* words to the right and left of the keyword. If a punctuation mark is reached before *n*, then the context window will be shorter. Thus, with *n* =6, context windows are a maximum of 13-word chunks. Context windows can be selected as words or sentences; i.e., a user can extract all keyword-containing sentences and the *n* sentences to the left and right of a keyword.

Table C.1: Illustrative extracted context windows with fictional company names

id	firm	text
10D4977B-000	Acme Inc. 1	{you shall not hire or permit ...}
ADSPO15-0807	Acme Inc. 2	{will not hire an applicant...}
CHR21009-CHS	Acme Inc. 3	{a contractor shall not employ ....}
99999-001	Acme Inc. 4	{non solicitation agreements with ...}

Note that for display purposes, only outputs relatively short in length were chosen. It is often the case that the extracted "text" contains multiple chunks extracted per document. The output of the extract algorithm is an "intermediate" dataset of (ID, text) tuples, where the text is simply a long '|' delimited string of all the extracted context windows. In the example shown, one can see that in each extracted chunk of text, a keyword is present, along with its extracted context. The first step towards building up rule sets is extracting all of the plausibly relevant text. Exploration of n-gram structures within the extracted data ensures that the context window is correct. This is accomplished with the algorithm presented in Algorithm C.2. The result with a given *n* is a file with the enumerated occurrences of each relevant n-gram, sorted in descending order.

A researcher is now enabled to obtain a general picture of what context windows appear with the unstructured text data, as well as their relative occurrences. Using this, the researcher follows

two decision processes: (1) the creation of a tag set, and subsequently (2) the creation of a rule set for each tag. Both sets are defined below.

Algorithm C.2: N-gram Exploration Pseudocode

```
Require: Parent Directory: where the extracted CSVs are located, n: desired n-gram size
Ensure: file with enumerated n-grams
global_counts = {}
for file in parent directory do
    for row text in file do
        split = text.split('|')
        for chunk in split do
            length = len(chunk.split())
            rules = chunk.split()[max(0, length/2 - n), min(length, length/2 + n)]
            global_counts.update(Counter(rules))
return global_counts.to_csv()
```

A tag is a title given to a certain characteristic of a particular document. These tags can be easily and simply defined, and are binary in nature, i.e., 1 if true or 0 if not true. The collection of defined tags makes up the tag set. Each tag in the set is assigned either 1 or 0 for every unique document in the original data. For the franchise document example, the following tag is defined:

- *nopoach* – a non-solicitation clause that prohibits one franchise from soliciting or employing any worker from another franchise.

One can define an arbitrary number of tags; furthermore, exactly what these tags mean, i.e., what definition they take on, is entirely up to the researcher. Binary classifications can be further broken down by defining additional "sub-tags." An examination of the context surrounding non-solicitation suggested further tags for future research. At the data analysis stage, these can be combined to create new measurements of concepts involving multiple tags. Crucially, the meaning that is assumed by each tag is defined via its rule set.

For each given tag, a researcher defines a list of rules that encompasses the definition and characteristics of this tag. More concretely, a rule is a chunk of any number of words. Therefore, a rule can be a single word, or even an entire sentence (up to the maximum length of the

extracted chunks, here 13). A single rule set, defined within a corresponding file, can contain one or more tags. If more than one tag is contained within a rule set, this means the tags that are bundled together are related enough such that some rules may overlap, or some rules may define one tag, while precluding the other. For each tag contained within a rule set, this tag is defined for each given rule, i.e., is assigned either 1 or 0. Finally, every rule in all rule sets must be assigned a priority ("prio"), which indicates its relative order of operation compared to other rules within the same rule set. An example excerpt from a rule file is given in Table C.2.

Table C.2: Excerpt from the rule set of the nopoach tag in formative stages

rule	prio	nopoach
hire	0	0
shall not hire	1	1
will not hire	1	1
may not hire	1	1
you shall not hire or permit any third party or outside vendors to access or perform any service	2	0
may not hire an applicant who has a felony	2	0
will not hire any person regardless of medical marijuana card	2	0
shall not hire or promote anyone who may have contact with residents	2	0
not hire offer to hire or otherwise solicit any employee	3	1

Priority is important in the sense that it defines a hierarchy of how text chunks are assigned certain tags. From the above example, one starts with every chunk containing "hire" to be *nopoach*=0 at *prio*=0. When, however, one encounters "may not hire", this then is a plausible indicator that *nopoach*=1, and a subsequent rule "may not hire" assigned priority *prio*=1 proceeds to classify it with *nopoach*=1. Upon examination, that rule turns out to have exceptions. Moving on to a higher priority rule, *prio*=2 can create exceptions to *prio*=1 rules: for example, with *prio*=2, a rule that states "may not hire an applicant who has a felony" overwrites the previous tag allocation, now assigning text chunks containing the string of text in the rule to *nopoach*=0. In this way, priority is important to handling iterative coding work and possibly overlapping or contradicting rules.

The final important aspect of rule creation comes with the use of Regular Expressions, denoted by rules with the "REGEX:::" qualifier prepended to the regular expression itself. Instead of simple text matching for a rule, regex rules verify if a particular text chunk satisfies the regular expression or not. Their usage allows for generalization, in situations where contexts are variable in length and content, yet possess the same general meaning. Once the researcher feels that the rule set is saturated, it is important to "prune" rules and validate them through repeated extrapolation and adjustment. Context windows should be no longer than the longest rule, which should be no longer than necessary to precisely capture the tag.

Extrapolation: Crossing the Bridge

With this collection of rule sets (for all defined tags) in hand, a tool is now needed to translate the rules contained within to a workable dataset. Such a tool is useful because it takes the manually coded context rules as input, and it then proceeds to "extrapolate" them into a training dataset from either a selected subset of extracted data files or the full dataset. Thus, we can cross the bridge from expert-created rules to structured, annotated datasets.

The Subset As mentioned, the extrapolation algorithm can operate on a selected subset of the entirety of the data, so as to create a representative training set, from which a robust, accurate classifier can be trained. This is accomplished by randomly and uniformly selecting documents from the original text files. In the case where a model will not be learned, i.e., where only the extrapolated dataset is desired, performing this sampling is not necessary.

Algorithm C.3 creates a training data set for each tag. Note that this algorithm is run per tag, meaning that the eventual output is a training set for each tag, which will then be used to train a classifier for each tag.

Algorithm C.3: Extrapolation Pseudocode

Require: Parent Directory: where the subset CSV files are located, **Rules File**: a rule set for a given tag or tags, **s**: sampling rate, **do_neg**: whether to perform negative sampling
Ensure: training data set for given tag(s)

```
1: results = []
2: for file f in subset do
3:     data = read_csv(f).sample(s)                          ▷ sample desired fraction of data
4:     for text in data do ▷ each '|' delimted extracted line
5:         for rule r in rule set do
6:             match = []
7:             plus = 0 ▷ for negative sampling only
8:             for x in text.split('|') do
9:                 if 'REGEX' in r then
10:                    if regex.search(r, x) then
11:                        match.append((x,1))
12:                        plus += 1
13:                    else if plus > 0 and do_neg == True then
14:                        if not any ru in x for ru in rule set then
15:                            match.append((x,0))
16:                            plus -= len(tags)                ▷ i.e. number of tags in rule set
17:                else
18:                    if r in x then
19                         match.append((x,1))
20:                        plus += 1
21:                    else if plus > 0 and do_neg == True then
22:                        if not any ru in x for ru in rule set then
23:                            match.append((x,0))
24:                            plus -= len(tags)
25:            random.shuffle(match)
26:                for m in match do                          ▷ m[0] = text, m[1] = negative sample?
27:                results.append(text chunk, rule, priority-weighted length, tag encoding)
28: training = DataFrame(results)
29: training = training.sort(priority-weighted length).drop_duplicates(chunk)
30: return training as CSV
```

One important note about the extraction algorithm is the sampling rate. Since many files are included in the subset, running extrapolation on the entirety of this subset would result in quite large training datasets. To avoid this, only a random sampling of each subset file is taken, in order to ensure a manageable training dataset. Secondly, the concept of negative sampling is incorporated into the algorithm. Such a concept allows for "negative", non-keyword containing text chunks also to be included in the training data, in the case that a classifier that can discriminate between keyword- and non-keyword-containing instances is desired.

Ultimately, the main purpose of the extrapolation algorithm is to bridge the human-centric rule creation phase with the ensuing model training and classification stage. Concretely, the manually created rule sets that are the product of the former are converted to large, yet workable training data sets that are vital to the functioning of the latter. Thus, this transition from defined rule sets to classifier training is facilitated by such an extrapolation method.

As described above, a user may need to augment the training data produced from CRAML in order to improve model performance. Poorly performing models may reflect that the underlying tags are not well-defined, and need additional refinement, or simply that there are not enough positive or negative cases in the corpus to build a reliable classification model.

After an initial extrapolation, the expert engages in iterative testing to revise the rules to achieve a comprehensive and accurately coded dataset. The expert does this with output from the extrapolation. An excerpt (shortened for readability) of the *nopoach* rules is provided in Table 5. The extrapolated training data files contain the extracted text chunks, which rule "caught" the particular chunk, and finally the appropriate tag value. This file can be reviewed by the user to determine if the rules are working as intended. A sample of chunks that oversamples positive tags (=1) and provides a minimum number of cases per rule can also be sent at this point for blind review by third-party coders to validate and ensure inter-subject reliability.

In Table C.3, early rules such as "shall not recruit", "shall not employ", and "shall not hire" were later over-written due to the multiple exceptions to these rules. Instead, statements that clearly stated that *employees* shall not be recruited or solicited or hired were tagged *nopoach*=1.

Table C.3: Sample entries from the novel training data created for nopoach during testing

id	chunk	rule	nopoach
527557	shall not recruit or hire any employee or former employee offranchisor or any	shall not recruit	1
272571	in item ahove you may not solicit customers from outside your territory without	may not solicit customers	0
487216	or our designee you will not hire third party or outside vendors to	you will not hire third party or outside vendors	0
792020	franchise lyou may not recruit or hire any employee or former employee of	may not recruit	1
714298	non solicitation of employees employee agrees that during	non solicitation of employees	1

Validation: Involving Third-Party Independent Coders

Validation helps to ensure inter-rater reliability, accuracy, and precision. Involving third parties at the stage at which rule sets are producing seemingly reliable results tests whether the tags are well enough defined to be agreed to by third parties. The researcher gives an independent coder only a description of the desired tag, the text chunks, and deletes the rules and the CRAML-generated coding. The independent coder completes their review, and the researcher compares the results of the rule set produced coding with the independent coder's coding. All discrepancies should be reconciled before proceeding further.

Based upon the performance displayed by this last step in the process, or by an externally imposed requirement to repeat the process, the researcher can choose to revisit the tags and rule sets, once again performing an exploration to search for more representative rules. It is important to emphasize the necessity of judgement in this phase, as the decision to repeat this process is a subjective one. Incorporating hand-coding by independent research assistants can help greatly. Moreover, the encoding, i.e., tag values, given to each particular rule must come from knowledgeable, grounded reasoning, especially in cases where certain keywords serve different meanings in possibly very disparate contexts. Therefore, it is in this cycle where the most manual effort is required, but also where the crucial foundation to the rest of the framework is built.

Model Training and Classification: Datasets in Action

The final stage in the proposed framework is the classification itself, i.e., the mapping of a particular text chunk to its corresponding tag set encoding (0/1 for each tag). In order to accomplish this, classification models must be learned from the rule sets discussed in the previous section. The presence of context rules alone is not sufficient in the sense that they represent high precision classification rules that will always detect exactly what is described by the rules. The next step of using a classifier is to learn a model that not only detects these "simple" cases that can be caught by string matching or regular expressions, but rather one that can also learn in which *general* contexts keywords appear which cause a certain tag to be true. This motivates the need for comprehensive training data.

The final step in the testing cycle involves the training of a baseline classifier to test performance on an unseen test set. In this case, a simple Naive Bayes classifier was chosen. First, training instances are converted to TF-IDF vectors, and then a Naive Bayes classifier is trained for each tag. Finally, output metrics are displayed, namely Accuracy, Precision, Recall, and F1. Using these metrics, a researcher can (roughly) evaluate the current performance of the classifier, which indicates the strength ("representativeness") of the underlying training data, i.e., rules.

Flexibility in Machine Learning Methods

An important step towards the building of a general-purpose classification system was to identify the Machine Learning method best suited for the multi-label, binary classification task at hand. Crucial to note is this multi-label aspect, as it is certainly possible for a single document to have more than one tag attribute be present. As such, multiple candidates for classification models were chosen, all of which could handle this multi-label binary classification task. So far, only "shallow" Machine Learning models were tested. The methods included:

- Naïve Bayes –also used as the baseline classification method. Simple probabilistic classifiers using Bayes' theorem as a backbone. Relatively easy and efficient to train.
- Logistic Regression – classification technique utilizing the logistic function (and a classification threshold) to predict the value of a dependent variable.
- Stochastic Gradient Descent Classification – a misnomer in the sense that SGD does not actually perform the classification. Rather, SGD is used to optimize a linear model, in this case a Support Vector Machine.
- Random Forest – a classifier using an ensemble of Decision Trees.

In a test of the various methods on a sample classification task (for the *nopoach* tag), Table C.4 displays the performance of these classifiers. Precision measures the percentage of correctly identified positive cases, i.e. how many of the classified '1's are indeed truly positive cases. Recall measures that percentage of correctly identified positive cases among all positive cases in the true labels. Together, these two metrics can be summarized in the F1-score, which is the harmonic mean of the two. As can be observed, the Random Forest model shows superior performance and was chosen to be the classification method.

Table C.4: Performance of various ML classifiers

	Acc.	Precision	Recall	F1
Naive Bayes	0.93	0.60	0.95	0.74
Logistic Regression	0.99	0.91	0.96	0.94
SGD SVM	0.98	0.91	0.92	0.92
Random Forest	**0.99**	**0.97**	**0.96**	**0.97**

With the respect to the general-purpose nature of this text classification framework, it is important to emphasize that the choice of classification method here will not necessarily be the optimal choice for other applications. In addition to this, the utilization of more advanced and powerful methods could certainly prove to be beneficial, and this remains a topic for future

research. In the end, though, the framework allows for essentially any model to be used, if it can be properly stored and loaded for use in the process pipeline. The task of choosing a specific model is left to the user. This make sense because different data, different domains, and different desired outcomes may require different models to be chosen. In the end, the focus of CRAML on the data *creation* process allows for a flexibility in the choice of model to train on this data.

Outline of Model Training

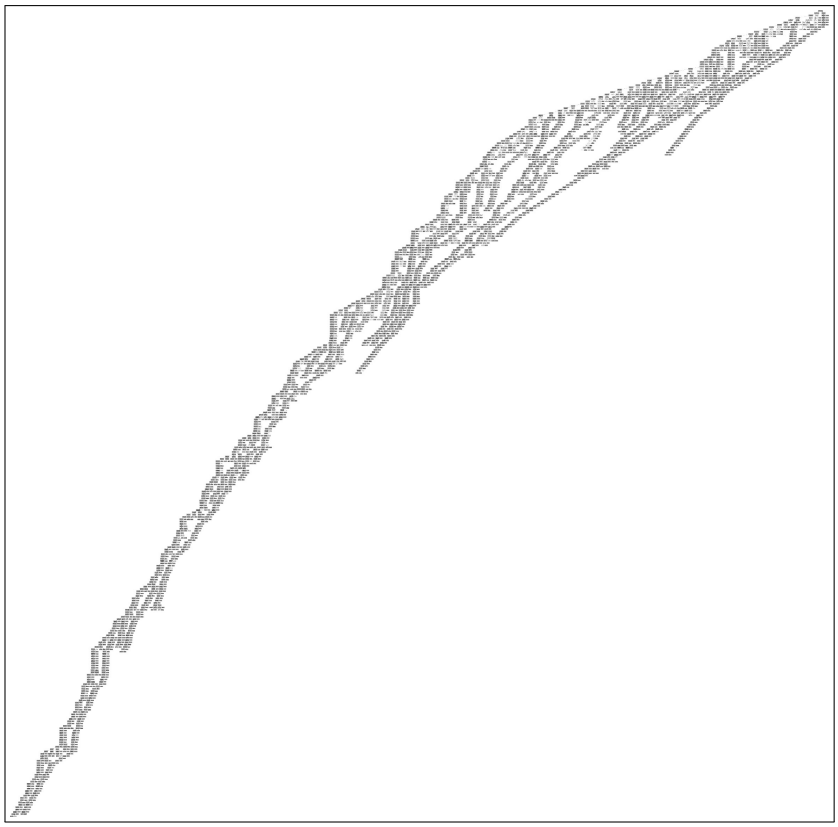
In order to adapt the use of classification models such as Random Forests for the classification of novel text datasets such as the one described throughout the previous sections, TF-IDF is utilized to convert a database of unstructured text entities into a matrix of numerically valued vectors. Concretely, for a dataset containing n documents and a vocabulary of m words, the resulting TF-IDF matrix is n x m in dimension. Through this vectorization of text, models can be trained and classification can now be performed. As a point for future work, the utilization of more advanced (and potentially meaningful) text representations could lead to richer training data for the proposed CRAML framework.

Training - Grid Search With the training of ML models comes many tuneable parameters. In order to optimize the resulting classifier for each tag, a tuning stage is added tp determine the optimal parameters, e.g., for a Random Forest, given a particular dataset. Here it is important to emphasize the significance of this stage in the overall framework. As all potential incoming datasets may be different in nature, different parameter values may be needed to achieve the best performance possible. In order to best support the general-purpose nature of the proposed framework, performing fine-tuning of parameters is crucial.

Training - Purification Prior to creating the classifiers for each tag, a purification process was run on the trained models, in order to reduce size, and as a result, classification time. This

eliminates unneeded dependencies and in the case of Random Forests, prunes the trees according to certain criteria. The effects of this process were quite dramatic, reducing most models to at least half the size. The potential complexity of Random Forests, and thus the need for the purification process, is visualized in Figure C.1, which display a pre-purification decision tree.

Figure C.1: One of the decisions trees within the nopoach Random Forest



Once a model for each tag is trained, it is then stored in pickle format, so that it can be later loaded and utilized within the framework. To do this, the user must manually enter the filename of each saved model into a JSON formatted file, which maps each tag to its corresponding classifier. Note how these tags match up with the mapping of tags to manually defined keywords. The naming convention used for the saved models was to specify the method used, the tag to be classified, the achieved F1-score, and the use of purification or not.

Negative sampling (optional)

Negative sampling was left as an option in the extrapolation process (Algorithm C.3). This decision would most directly affect the function of the classifiers. Without negative sampling, the classifier for each tag would be to "classify a tag as 0 or 1, provided that the chunk to be classified contains a tag-defined keyword." On the other hand, performing negative sampling process introduces random, non-keyword-containing text chunks into the model training process, thereby learning these instances into the given model. Here, the classification task becomes "classify any given piece of text as 0 or 1, regardless of its content". With these two approaches in mind, it was ultimately decided to implement the former, thereby only advancing a text chunk to the classification stage if it indeed contained a keyword. If not, the classification was automatically made to be 0.

This design decision was made knowing the makeup of the textual data, as well as the characteristics of the tags to be classified. In the end, in order for a tag to be assigned a 1, a necessary condition is for it to contain a tag-specific keyword. Otherwise, it is not possible to obtain the classification of 1. The classification process, therefore, is in place to "weed out" keyword-containing text chunks that do not lead to the desired tag attribute. In other domains, i.e., with other text datasets created via this framework, this line of reasoning may not hold, and an argument could be made for the use of negative sampling. In this way, it is once again paramount for researchers to define their tags thoroughly and explore the makeup of the underlying data.

Pre-Processing (optional)

One final component is the option of a pre-processing step for each classifier. The motivation is that in the text chunks, which are relatively large in length compared to the single keyword

within, there might exist much information, i.e., words, that do not play a role in a particular tag at hand. Particularly with tags in which the pertinent information sits close to the keyword, any other irrelevant information will only serve to confuse the learning of a classification model. With this in mind, the proposed pre-processing step will serve to "trim" the left and right ends of a given text chunk, thus reducing the amount of text noise within this chunk. In this process, though, two considerations must be made. Firstly, if certain qualifier words, such as negations, exists within the regions to be trimmed, this could potentially be vital information lost. Secondly, there may be specific known words or phrases that appear that usually appear in the proximity of a keyword, yet might also be trimmed away. For these reasons, two more mappings, following the structure of previous ones where an entry exists for each tag, are defined. Qualifiers involves the words that have the ability to change the meaning of a text chunk in a significant manner. Keep words are just that – words that should be kept, even though they exist in a region to be trimmed away.

Algorithm C.4 outlines pseudocode for pre-processing steps. In the testing with this preprocessing stage, the TRIM parameter was chosen to be 2, resulting in chunks of length 5, not including possibly kept words. In the end, this stage was excluded because there was an observed decline in classification performance for many tags. This sheds light on the importance of relatively faraway context words from the main keyword for many of the tags. Again, this may differ with other datasets, so testing with a preprocessing stage is recommended.

Storing

The final process in the framework is to consolidate the classification results, merging them with the original data to be stored for later retrieval and analysis. The framework facilitates for the storing of all classification outputs to CSV or a database of choice.

Algorithm C.4: Preprocessing Pseudocode

Require: Training CSV: file that contains the training data to be trimmed, **Tags**: tags to be processed, **Qualifiers**: for each tag, **Keeps**: for each tag, **TRIM**: target context size
Ensure: trimmed training data

```
new_data = []
for text in training data do
    index = get_index(text)                          ▷ get index of keyword within chunk
    for q in Qualifiers do
        found = search_for(q, text)                  ▷ find qualifiers in chunk
        if found then
            text = qual_prepend(found, text)         ▷ prepend qualifiers to words in chunk
    lower = max(0, index - TRIM)
    upper = min(len(text), index + TRIM)
    new_text = text.split()[lower:upper]
    keep = []
    for k in Keeps do
        if k in text then keep.append(k)
    new_text = keep + new_text
    new_data.append(new_text)
return new_data
```