Information Extraction from Materials Science Literature: Progress in Machine Learning Approaches

by

Haihao Liu

Bachelor of Science in Materials Science and NanoEngineering
Bachelor of Arts, Mathematics
Rice University, 2018

Submitted to the Department of Materials Science and Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Materials Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2024

Signature of Author ____
Department of Materials Science and Engineering
June xx, 2024

Certified by ____
Elsa A. Olivetti
Associate Dean of Engineering, Jerry McAfee (1940) Professor in Engineering,
Professor of Materials Science and Engineering, MacVicar Faculty Fellow
Thesis Supervisor

Accepted by ____
Robert J. Macfarlane
Associate Professor of Materials Science and Engineering
Chair, Departmental Committee on Graduate Studies

Information Extraction from Materials Science Literature: Progress in Machine Learning Approaches

by

Haihao Liu

Submitted to the Department of Materials Science and Engineering
on June xx, 2024 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Materials Science and Engineering

Abstract

Materials informatics is an emerging field that uses data-driven methods to accelerate materials design and synthesis. With this development, there is a concomitant need for an ever-increasing amount of data. Yet, there is a paucity in the availability of large, high-quality datasets on experimentally-obtained materials data, compared to other fields like biomedical. A huge wealth of information remains untapped and not directly accessible to machine learning algorithms, in the form of articles published in peer-reviewed academic journals. This is because the data is presented in natural language text or tables intended for humans to read, rather than a computer-friendly format. We define and address several tasks, that are to varying degrees specific to materials science, all falling under the general problem of extracting structured information from unstructured and semi-structured text in published literature.

Thesis Supervisor: Elsa A. Olivetti
Titles: Associate Dean of Engineering, Jerry McAfee (1940) Professor in Engineering, Professor of Materials Science and Engineering, MacVicar Faculty Fellow

Acknowledgements

Elsa

Frances, Suraiya

Chiajou, Iris, Lisa, Tony

Friends, esp. Kevin

Olivetti Group: Zach, Alex, Olivia, Kevin, Rubayyat, Vineeth, Mrigi; Tuna, John, Hugo, Xinkai, Brian, Basuhi, Jacqueline, Luca, Brooks, Chris, Elton

Collaborators: UMA – Tim, Sheshera; Novelis

Family, Timmy renewed hope

List of Figures

1.

List of Tables

1.

Contents

Abstract 4

Acknowledgements 6

List of Figures 7

List of Tables 8

1 Introduction 11

2 Background 12

2.1 Applied Machine Learning 12

2.2 Materials Informatics 12

2.3 Information Extraction 12

2.4 Natural Language Processing 14

2.4.1 Empirical linguistics 14

2.4.2 Contextualized embeddings 15

3 Key Phrase-based Abstracts Analysis 16

3.1 Task Definition 16

3.2 Methods 17

3.2.1 Abstracts database 17

3.2.2 Key phrase ranking 18

3.2.3 Topic co-occurrence 20

3.3 Results and Discussion 20

3.3.1 Top key phrases in corpus 20

3.3.2 Top key phrases by series 21

3.3.3 Top co-occurring topics 21

3.4 Future Work 21

4 Aluminum Alloy Data Extraction **22**

4.1 Task Definition 22

4.2 Methods....23

4.2.1 Data sources 23

4.2.2 Dataset preparation 24

4.3 Results and Discussion 24

4.3.1 Composition data 24

4.3.2 Properties data....25

4.3.3 Data visualization....25

4.4 Future Work 27

5 Sample Name Recognition **29**

5.1 Task Definition 29

5.2 Methods....30

5.2.1 Dataset preparation 30

5.2.2 Model architectures....31

5.3 Experiments and Discussion 33

5.3.1 Training....33

5.3.2 Results....34

5.3.3 Explainable classifier 36

5.4 Future Work 36

6 Outlook and Conclusions **38**

1 Introduction

Of central importance in materials science is the idea of the structure–process–property relationship. In fact, much of the discipline can be understood in terms of studying their interplay and codependence in various material systems. There is a vast wealth of information contained in the literature investigating these relationships, with lots of data embedded as unstructured text within journal articles. It would be of great use to researchers if this data could be extracted into a structured format suitable for further computational or manual analysis.

When human researchers perform a literature search and review, we might approach journal articles from several different angles depending on what sort of information we are looking to get from them. Generally, we would start by reading the abstracts to get some idea of the areas of research being done, types of methods being investigated, or classes of materials being studied. From there, we would delve deeper into some of the papers, looking at their quantitative results and the qualitative conclusions they draw. As part of that, we look at different sections of the paper and naturally and instinctively form links between disparate data referring to the same sample or experiment, to mentally form structure–process–property relationships.

Of course, human researchers are greatly limited in the number of papers they read in such a manner, limited by both time and the rate of processing information. Machines, on the other hand, face no such limitations, and will happily "read" millions of journal articles should one tell it to do so. The challenge, of course, is that research papers are not written for computers, who first have to "learn" what information to extract and how—hence *machine learning*. Any nuance that a computer would miss or mistake it would make, compared to a human, is more than made up for by the overwhelmingly vast increase in the volume of data at our disposal that such an approach makes available.

In this thesis, we define and address three information extraction tasks that roughly parallel how a human may approach reading a paper, in a comprehensive manner, as outlined earlier:

1. Key phrase ranking and analysis, from abstracts alone
2. Aluminum alloy data extraction, from tables and body text
3. Sample name recognition, in order to link separate pieces of data

Various approaches and methods will be used to tackle these, as is necessarily the case for such divergent tasks. Different methods and models will also be compared within each task. The overarching goal is to make the data available for use to the wider materials research community, but more importantly, develop and disseminate information extraction methods that are even more broadly applicable in fields outside materials science.

2 Background

2.1 Applied Machine Learning

Even before the advent of neural network models, systems to predict physical properties from material composition were developed, for example the Materials Algorithms Project at Cambridge [1]. As neural nets gained popularity, this prediction task became a popular use of machine learning since it is such a straightforward and classic application of a basic neural net. It takes in percentages of a fixed number of elements as input (one node for each), and produces a single number as the prediction for some property as output. There are challenges with this however, such as difficulty extrapolating, and being a completely uninterpretable black-box model. Some research is being done to address this, to varying degrees of success [2].

While we do now use more advanced models, and although more and more materials scientists are starting to use ML for research, very often they only use the most basic models in production-ready tools like `scikit-learn`, particularly in industry. The fields of organic chemistry [3], biology [4], medicine [5], and fluid mechanics [6], among others, are more advanced in this respect, having already begun to incorporate their domain knowledge into statistical and ML models. In order to avoid hitting a plateau, there needs to be more crosstalk between the materials science and computer science communities, because we have reached a point where more specialized and application-tailored approaches are needed to make progress [7], [8].

2.2 Materials Informatics

Over the past half-decade, tremendous strides have been made in the field of materials informatics, growing from a niche subject to a recognized subdiscipline of materials science [9]. Broadly speaking, materials informatics refers to the use of data and algorithms in materials research, which can be seen as a "fourth paradigm" of scientific inquiry, after experiments, theory, and simulation [10]. Meanwhile, computational materials science, itself having huge success in the past ten years or so, mainly refers to density functional theory (DFT), molecular dynamics, phase field modeling, and other physics-based tools, and should be distinguished from the data-based approach of materials informatics [11].

For the most part, modern research using ML in materials science has largely been for materials discovery and design [31]–[33]. One of the first large-scale efforts on this was the Materials Project, which leveraged high-throughput DFT, and later ML, to build a database of hypothetical materials and their predicted properties based on structure [34]. Materials with potential applications in batteries [35], hydrogen storage [36], and catalysis [37] have all been identified, just to name a few examples. Now the bottleneck lies in actually synthesizing these materials. This is because far less work has been done applying ML to materials synthesis and processing. Prior work in our group has investigated the synthesis of metal oxides [38], [39], zeolites [40], and perovskites [41].

2.3 Information Extraction

The central motivation behind the work in this thesis is that data is a key ingredient in machine learning, for materials or otherwise. In particular, we need *structured* data for materials informatics, that is, data in a form amenable to being processed by and used as input for computer algorithms. This leads to a straightforward and concise definition of information extraction (IE): the process of turning unstructured data into structured data [12]. The unstructured data input in this case can take many forms, but generally speaking all fall under the category of natural language in textual i.e., written form. The form of the output structured data is likewise varied, but a common feature is that there are fields and attributes with which to label and classify information, usually presentable in "informationally equivalent tablelike" or "machine readable" forms, called information formats in [13].

A brief history of IE provides the necessary context behind our work and some insight into the status of the field. This overview is drawn largely from [14] and [15], where Wilks first formally defines IE as a "core language technology". As early as 1964, the idea of "text searching with templates" was proposed by Wilks, although technologically infeasible at the time [16]. In the late 1970s, Sager successfully demonstrated what she termed "information formatting" of medical texts such as radiology reports and clinic narrative [13]. This was a highly effective application of the LSP computer grammar of English developed at NYU for natural language processing. Around the same time, work was being done by DeJong developing FRUMP to extract information on terrorist and other events from AP newswires [17], and Cowie on extracting canonical structures (flower color, shape, size, etc.) from field guides on flora and fauna [18].

These early systems had several limitations: there was a high degree of handcrafting in creating the structures, no access to external corpora for training, and use of no machine learning algorithms. In the mid 1990s

- ARPA-sponsored MUC competitions
- Started around 1997 (Google PageRank 1996, prototype 1998)
- International summer school on IE
- What Wilks calls "templates"
- Limitation: to what degree *is* knowledge template-like?
- The product of an IE system—the filled templates—can be seen either as a compressed, or summarized, text itself, or as a form of database (with the fillers of the template slots corresponding to conventional database fields).
- "One can then imagine new, learning, techniques like data mining being done as a subsequent stage on the results of IE itself."
- "One moral from all this … is that most of our cultural, political and business patrimony is still bound up with texts … The text world is vast and growing exponentially: one should never be seduced by multi-media fun into thinking that text and how to deal with it, how to extract its content, is going to go away.

Modern tasks

- Relation extraction → knowledge graphs (CS graph completion)
- Event extraction, event coreference, temporal expression
- Template filling
- This thesis proposes and tackles three new tasks

2.4 Natural Language Processing

Text is unstructured and semi-structured data described by free-flowing natural language that is not readily interpretable by machines

Manual data extraction is expensive, labor-intensive, and error-prone

2.4.1 Empirical linguistics

- Less on theory (formal grammar à la Chomsky)
- Late 90s revival of machine learning
- Modularity: "the notion that computational linguistic tasks could be separated out and evaluated quite separately, and that the separation between the modules need not correspond only to classic divisions of linguistic levels, such as syntax and semantics."
- Combine "intermediate tasks" (e.g., POS tagging) to perform large-scale tasks (IE, MT)
- EMNLP since 1996 https://aclanthology.org/sigs/sigdat/
- Automatic Content Extraction (ACE) evaluations of 2000-2007 on named entity recognition, relation extraction, and temporal ex- pressions1, the KBP (Knowledge Base Population) evaluations

Chemistry domain:

- Domain-specific terminology, numeric phrases
- CHEMDNER corpus
- ChemTagger: Hawizy et al 2011 ChemSpot: Rocktäschel et al 2012

Polymer domain:

- Tchoua et al, IEEE, 2017 DOI 10.1109/eScience.2017.23

Biology/medical domain:

- Publicly available annotated collections GENIA corpus
- Unified Medical Language System
- Riedel and McCallum, Empirical methods in NLP 2011

Continuum of text mining approaches

- Failed reactions database: Nature, **533**, 2016

- ChemDataExtractor: J Chem Inf Model, **56**, 2016
- LeadMine: Lowe and Sayle, J Chem Inf Model, **7**, 2015

- CNN: convolutional neural net
- HS-biGRU: half-stateful bidirectional gated recurrent unit
- FCN: fully-connected network
- CEM: chemical entity mention
- Korvigo et al., J Cheminform 2018
- Court and Cole., Scientific Data 2018

2.4.2 Contextualized embeddings

Contextual Prior to BERT, a major breakthrough in contextual representations was ELMo [3] (Embeddings from Language Models). ELMo was the first deep bidirectional language model, meaning it incorporates both left and right context of a word from the whole sentence, rather than just its immediate neighbors (i.e., shallow). Shortly after its release, Kim et al. fine-tuned the model with an unsupervised approach on a corpus of materials science literature [4], and made these domain-specific embeddings publicly available for the benefit of the materials community.

Kim et al. found that the ELMo embeddings gave better performance on several classification tasks compared to non-contextualized embeddings, namely word2vec and FastText, albeit at a greater computational cost. When BERT was released however, it set the new record on NLP tasks across the board. As a result, the community's attention quickly shifted towards using BERT instead for training tuned domain-specific models. Since then, several such models have been released, including SciBERT [5] (biomedical and CS corpora) and Clinical BERT [6]. Along the same lines, scispaCy [7] is version of the popular NLP package spaCy (used in this work) tailored to biomedical texts.

3 Key Phrase-based Abstracts Analysis

Key phrases, which are essentially multi-word "keywords", are the topics, concepts, or entities that are of central importance in a piece of text. Within the context of materials science literature, they can be, amongst many other things: a processing or synthesis method ("friction stir welding", "modified Hummers' method"), a measurable quantity ("fatigue crack growth rate", "initial specific capacity"), a qualitative outcome ("improved mechanical properties", "outstanding electrochemical performance"), or a physical process or proposed mechanism ("grain boundary sliding", "Na-ion diffusion"), to give a random sampling.

The goal of this chapter is to investigate methods of automatically extracting, ranking, and analyzing such key phrases. A first-order approach to extraction and scoring might be to use tf–idf, our baseline "zero" model in the chapter on sample names. This has two main limitations: first, it scores words one at a time, so we only get individual key*words*, not multi-word key *phrases*. Second, it needs a corpus to calculate IDF, so keywords cannot be extracted from just a single piece of text. Fortunately, there already exists a well-known and often-used algorithm that solves both of these limitations called Rapid Automatic Keyword Extraction, or RAKE [19].

In reading a paper, always start with abstract. By understanding extracted key phrases, one can begin to understand the status of a field as a whole: what are the biggest challenges, most common methods and materials being studied, etc., which can be especially useful for someone new to that field. A more advanced (albeit conceptually related) application is the construction of a "semantic network" or SemNet of co-occurring topics, which can then be used to predict research trends in a field [Mario PNAS 2020]. Taking this one step further, we can try to understand *relationships* between co-occurring topics, which one may call topic modeling [Kevin 2022?].

3.1 Task Definition

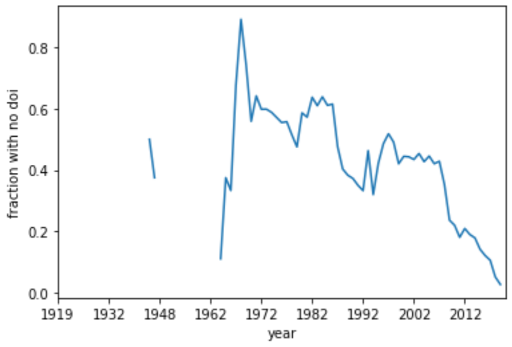
The information extraction task at hand is to automatically find the most important key phrases, as well as co-occurring pairs thereof, within abstracts and relevant sections of papers in a given corpus. Since RAKE already exists as a robust method to extract key phrases from an individual document, my first main contribution is in creating a scoring metric to help find the most important among them. A large subclass of key phrases can be regarded as "topics", and by determining their relative importance within a corpus, they allow us to synthesize or summarize the knowledge therein. An example of this on aluminum alloys will be presented, to supplement the results of the previous chapter.

My second innovation is a smart method to extract co-occurring pairs of key phrases, that is, both appearing in the same paragraph of a paper. In particular, co-occurrence within any one abstract in the corpus is what creates a link/edge between two key phrases, the nodes/vertices in a SemNet. The idea is, these topics must be important and related enough according researchers in the field, for them to appear together in the abstract of a paper. In addition to abstracts, we also look for co-occurrence in synthesis methods or "recipes" for the topic modeling task. The goal is to understand the context of extracted key phrases: Is it a general problem in field? Or a specific problem this method in the paper addresses? Having a model to predict how co-occurring topics are related is key to this. Some initial results will be presented on topic modeling in the domain of sodium-ion batteries.

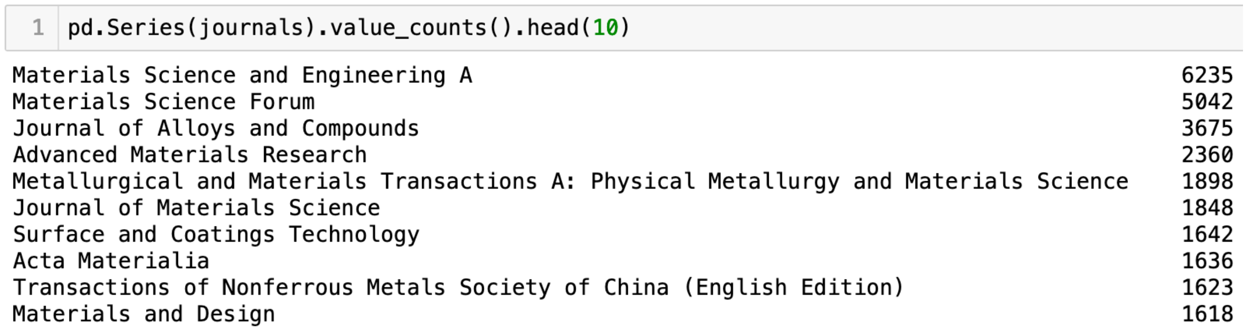
3.2 Methods

3.2.1 Abstracts database

For the aluminum alloy use case, we began by assembling a corpus of paper abstracts using Scopus, the world's largest abstract and citation database [cite]. This has several advantages. First, our papers database requires that the full text of articles be digitized, and so, with few exceptions, only goes as far back as around 1995, whereas far more older papers have had only their abstracts digitized, thus allowing us to go as far back as the early 1900s. Second, so long as we have access to Scopus, we have access to the text of abstracts in their entirety, even if due lack of subscriptions or licensing agreements we don't have access to the full text of the paper. Third, our papers database uses DOI (digital object identifier) as the unique identifier for every document within, but as it turns out, most papers published before 2000 (when DOI was introduced) have not yet been retroactively assigned one. Figure x shows the fraction of papers with no DOI over time.



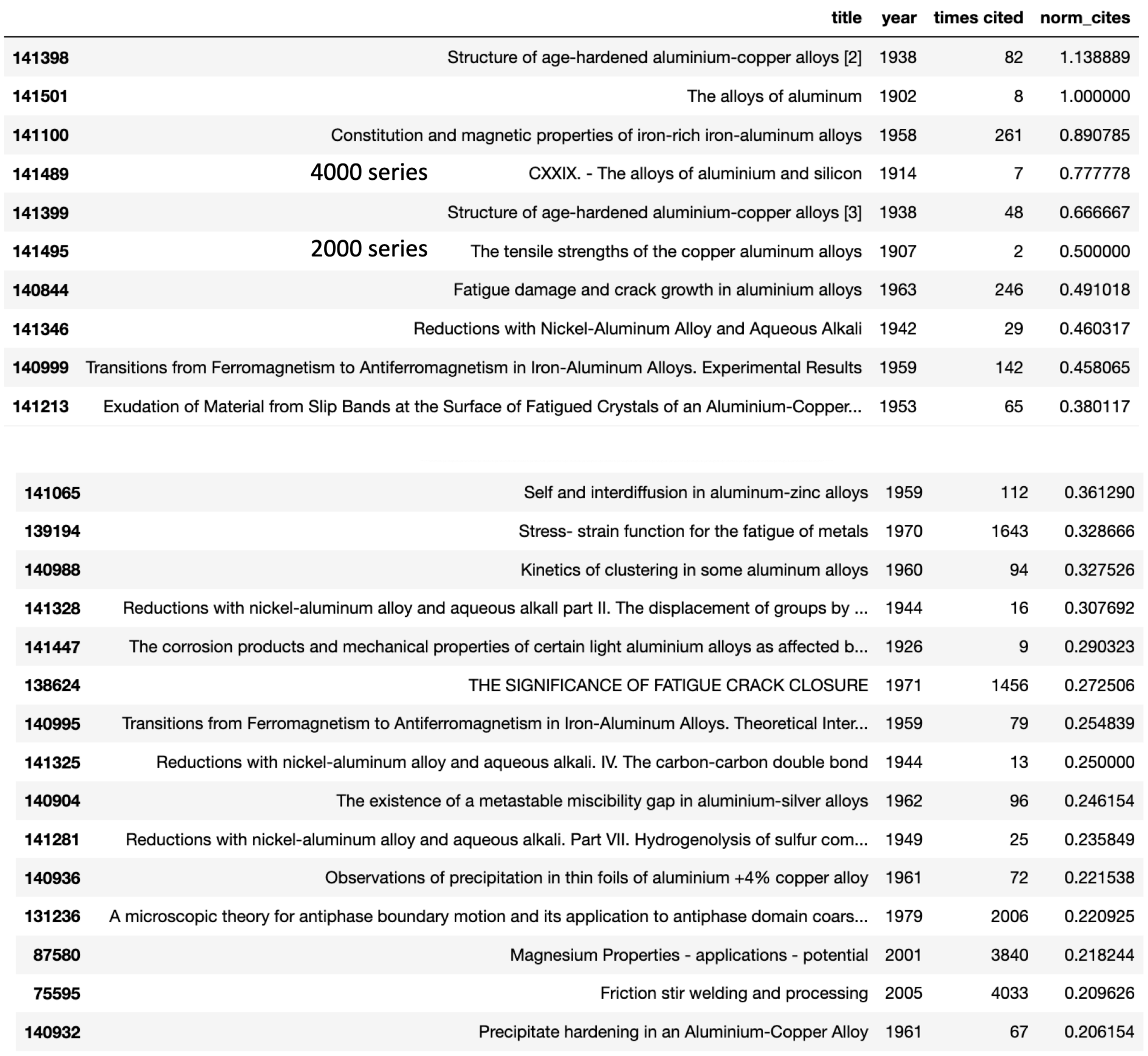
With all this in mind, we constructed the following query in Scopus: `TITLE(aluminum AND alloy) OR KEY("aluminum alloy" OR "aluminum and alloys" OR "aluminum metallography" OR "aluminum metallurgy" OR "aluminum sheet")`. This yielded a targeted set of 141,525 abstracts, of which 46,461 (33%) have no DOI, and 95,064 (67%) do. Table x shows the journals with the most articles published on aluminum alloys.



```
pd.Series(journals).value_counts().head(10)
```

```
Materials Science and Engineering A                                             6235
Materials Science Forum                                                         5042
Journal of Alloys and Compounds                                                 3675
Advanced Materials Research                                                     2360
Metallurgical and Materials Transactions A: Physical Metallurgy and Materials Science    1898
Journal of Materials Science                                                    1848
Surface and Coatings Technology                                                 1642
Acta Materialia                                                                 1636
Transactions of Nonferrous Metals Society of China (English Edition)            1623
Materials and Design                                                            1618
```

In order to get some idea of which are the most "important" papers in this corpus, we first looked at the number of citations, but quickly realized a problem: the top cited list strongly biases recent papers, since there is simply more research being published in general year after year, so naturally there would also be more citations. The solution was to "normalize" the citations by year, by dividing the number of citations a paper has by the total number of aluminum papers published in the subsequent 5 years, the rationale being that impactful papers tend to be highly cited shortly after publication. Table x shows the top papers by normalized citation count. What stands out is that it's able to rank papers from across the entire time period "fairly", exactly as intended. Papers from the 2000s with thousands of citations appear alongside papers from a century ago with only a few dozen citations or even fewer, indicating that we haven't overcorrected the bias. Moreover, we are able to identify some very early papers that later formed the basis for entire series of aluminum alloys, such as copper aluminum alloys from 1907 for the 2000 series, or silicon aluminum alloys from 1914 for the 4000 series.



	title	year	times cited	norm_cites
141398	Structure of age-hardened aluminium-copper alloys [2]	1938	82	1.138889
141501	The alloys of aluminum	1902	8	1.000000
141100	Constitution and magnetic properties of iron-rich iron-aluminum alloys	1958	261	0.890785
141489	4000 series CXXIX. - The alloys of aluminium and silicon	1914	7	0.777778
141399	Structure of age-hardened aluminium-copper alloys [3]	1938	48	0.666667
141495	2000 series The tensile strengths of the copper aluminum alloys	1907	2	0.500000
140844	Fatigue damage and crack growth in aluminium alloys	1963	246	0.491018
141346	Reductions with Nickel-Aluminum Alloy and Aqueous Alkali	1942	29	0.460317
140999	Transitions from Ferromagnetism to Antiferromagnetism in Iron-Aluminum Alloys. Experimental Results	1959	142	0.458065
141213	Exudation of Material from Slip Bands at the Surface of Fatigued Crystals of an Aluminium-Copper...	1953	65	0.380117
141065	Self and interdiffusion in aluminum-zinc alloys	1959	112	0.361290
139194	Stress- strain function for the fatigue of metals	1970	1643	0.328666
140988	Kinetics of clustering in some aluminum alloys	1960	94	0.327526
141328	Reductions with nickel-aluminum alloy and aqueous alkall part II. The displacement of groups by ...	1944	16	0.307692
141447	The corrosion products and mechanical properties of certain light aluminium alloys as affected b...	1926	9	0.290323
138624	THE SIGNIFICANCE OF FATIGUE CRACK CLOSURE	1971	1456	0.272506
140995	Transitions from Ferromagnetism to Antiferromagnetism in Iron-Aluminum Alloys. Theoretical Inter...	1959	79	0.254839
141325	Reductions with nickel-aluminum alloy and aqueous alkali. IV. The carbon-carbon double bond	1944	13	0.250000
140904	The existence of a metastable miscibility gap in aluminium-silver alloys	1962	96	0.246154
141281	Reductions with nickel-aluminum alloy and aqueous alkali. Part VII. Hydrogenolysis of sulfur com...	1949	25	0.235849
140936	Observations of precipitation in thin foils of aluminium +4% copper alloy	1961	72	0.221538
131236	A microscopic theory for antiphase boundary motion and its application to antiphase domain coars...	1979	2006	0.220925
87580	Magnesium Properties - applications - potential	2001	3840	0.218244
75595	Friction stir welding and processing	2005	4033	0.209626
140932	Precipitate hardening in an Aluminium-Copper Alloy	1961	67	0.206154

3.2.2 *Key phrase ranking*

Though our contributions in this chapter are not in developing methods for the actual extraction per se of key phrases, only the ranking thereof, it is instructive to have at least a basic understanding of the RAKE algorithm. In particular, we are using the implementation of RAKE from the Natural Language Toolkit (NLTK) Python package [cite].

First, a predefined list of stop words (is, not, that, etc.) is removed, leaving so-called content words. We use the default stop words for English from NLTK, as well as all punctuation marks. Next, adjacent sequences of content words are grouped together into candidate phrases. A maximum phrase length of 4 was chosen based on [cite], but this was increased to 5 for the batteries use case, because it was found to be needed to capture certain . RAKE then creates a co-occurrence matrix, where each row shows how often a given content word appears in candidate phrases with each of the other content words. From this matrix, a score is calculated for each content word: either as its *degree* (sum of co-occurrence counts in its row), as its *frequency* (number of times it appears in the text), or as its *degree divided by frequency*. We elected to use degree-to-frequency ratio as the matric, as it favors longer, more specific phrases, whereas using word degree favors shorter, more general phrases [RAKE]. A score can also be assigned for each candidate phrase, as simply the sum of the scores of its content words. Finally, candidate phrases are ranked by their scores, and key phrases are taken to be the top *n* entries on that list. We took *n* to be the top one-third of the candidate list, as the original RAKE paper authors do.

In this description so far, key phrases are only ranked within a document. The RAKE authors do however also introduce several metrics for understanding the relative importance of each key phrase found within a corpus of many documents. They are all based on counting how many times each candidate/key phrase *k* appears in any document's list of candidates/key phrases across the corpus. The RAKE authors denote the former as the "referenced document frequency" of *k*, $rdf(k)$, and the latter as the "extracted document frequency" of *k*, $edf(k)$, with $rdf(k) \geq edf(k)$, by definition. In our implementation, only counts for key phrases appearing in at least two documents are recorded (i.e., $rdf(k)$, $edf(k) \geq 2$), since those appearing in only one are too specific to that document for us to meaningfully analyze. We also manually remove artifacts and noise (e.g., publisher name which Scopus includes in the text of abstracts) from the list of key phrases, such as `©, °, </, ltd, gmbh, society, ×`, etc.

With these counts, the RAKE authors define several scores for each key phrase *k*. The first is the *exclusivity* of *k*, $exc(k)$, a measure of how often a phrase was in the "exclusive" top third of candidates.

$$exc(k) \coloneqq \frac{edf(k)}{rdf(k)} \leq 1$$

An exclusivity score of 1 means the key phrase was extracted from every document in which it was a candidate. Some exclusive key phrases appear in more documents that others, indicating they are, in a sense, more "essential" to the corpus as a whole. The authors thus define the *essentiality* of *k*, $ess(k)$.

$$ess(k) \coloneqq edf(k) \times exc(k)$$

On the other hand, some key phrases can be considered "general" to the corpus, in that they are candidates in many documents but were only extracted as key phrases from a few. This intuition can be formalized by defining *generality* of *k*, $gen(k)$.

$$gen(k) \coloneqq rdf(k) \times \big(1 - exc(k)\big) = rdf(k) \times \left(1 - \frac{edf(k)}{rdf(k)}\right) = rdf(k) - edf(k)$$

With these metrics defined, and taking inspiration from tf–idf, a fourth metric is introduced in this work. It was noted that term frequency is in some sense a crude way of capturing the same idea as essentiality, while document frequency is likewise a proxy for generality. We thus substitute each with its counterpart and define the "pseudo-**tf**–**id**f" score of k, $tid(k)$.

$$tid(k) \coloneqq \frac{ess(k)}{gen(k) + \varepsilon} = \frac{edf(k) \times exc(k)}{rdf(k) - edf(k) + \varepsilon}$$

The ε term in the denominator serves two purposes. First, it prevents division by zero, since any k with $exc(k) = 1$ will have $gen(k) = 0$. Second, it acts as a tunable balancing term of sorts. The idea here is to consider two hypothetical key phrases, one that only appears in a few documents, say, $edf(k) < 10$, while the other appears in many but is also fairly general, with $gen(k) \approx 1$. Intuitively, we would want to weigh them about the same, since both types are probably interesting for us to look at. After some experimentation, and noting that the highest $ess(k)$ for the second type of key phrase is in the thousands, we settled on using $\varepsilon = 1000^{-1} = 0.001$. Adjusting ε allows one to favor one type of key phrase on this spectrum over the other.

Though we did Ranking by aluminum series

Small co-occurrence tests (top 100 = 10,000 pairs)

Rank all keywords in collection (by tid), choose top n = 100/1000/etc.

Search for all $O(n^2)$ possible pairs in every paper in collection (SLOW!)

3.2.3 Topic co-occurrence

Where in doc, where in para; when co-occur, order and proximity.

Create all $O(n^2)$ possible pairs with tracked locations from one paper ($n < 100$)

Rank pairs across collection afterwards (by frequency is easiest, or scores)

Already have locations from regex match earlier

Iterate over all $n(n-1)/2$ possible combinations, retaining order

Proximity is simply difference in two (average) match locations

Order is simply sign of proximity (convention: second – first)

3.3 Results and Discussion

Of the various forms of data above, composition data in particular is very high-dimensional in nature. This is because each constituent element is essentially its own dimension. In order to effectively analyze and visualize such a dataset, several dimensionality reduction techniques from data science can be used.

3.3.1 Top key phrases in corpus

Including small linked keywords test

3.3.2 Top key phrases by series

The Synthesis Project pipeline was built in Keras [20],

3.3.3 Top co-occurring topics

The Synthesis Project pipeline was built in Keras [20], so I will most likely also use that to maintain maximum compatibility. A common problem for many real-word datasets is class imbalance.

3.4 Future Work

Construct SEMNET for Al alloys

Topic modeling with Kevin

Comments from Novelis about 7000 series

4 Aluminum Alloy Data Extraction

After extracting as much qualitative information as possible from paper abstracts, both in the traditional sense, while reading papers during a literature review, or through a high-throughput machine learning approach like that presented in the previous chapter, we next aim to extract all the empirical quantitative data, be it experimental, calculated, or otherwise, from the main body of the paper. In this chapter, we present several machine learning approaches to accomplish this, some more successful than others, extracting from a variety of different formats and source document types. In particular, we will be focusing on the specific domain of aluminum alloy design, by way of application.

Researchers continue to explore and develop aluminum alloys with new compositions and improved performance characteristics. While this area of research remains active, the approach tends to be more traditional than with, say, electronic materials, especially outside academia in industry. In practice, alloy development generally starts with a manual literature search, and the data used is largely empirical. That said, the field of alloy design is gradually becoming more computer- and data-driven, for example with programs like the Obama administration's Materials Genome Initiative on alloys.

A fuller understanding of the current design space would greatly aid in and accelerate the discovery of new alloys, and machine learning can help us bridge that gap. In particular, an aluminum company seeking to gain a competitive advantage would need to explore far beyond the compositions registered with the Aluminum Association (AA), the governing body for aluminum standards globally. Despite the number of registered alloys increasing from 75 in 1954 to over 500 today, a much, much broader design space is actively being researched in academic laboratories and groups around the world.

Yet, there remains a critical lack of and need for structured data in a form amenable to materials informatics. There are few open databases on aluminum alloys, and of those, most only have a few computed properties on a limited number of compositions, and not experimental data on strength or ductility, due to the difficulty and computational cost to calculate those. The work in this chapter led to us producing and publishing the first (to our knowledge) open, extensive database of experimental data on aluminum alloy compositions and properties.

4.1 Task Definition

From the full body text of papers on aluminum alloys, which must first be identified, we will extract the following information:

a) Compositional information from table data as well as body text mentions of alloy series by name and composition. This will be done through a previously developed table extraction tool created by our group [21].

b) Phases resulting from those compositions. We will try to also track processing conditions although that will largely be out of scope for this initial examination.

c) An attempt will be made to extract yield strength, measures of ductility (and temper, where reported) associated with each document. We will attempt to extract these property values and make meaning of them.

4.2 Methods

Broadly speaking, the task of extracting aluminum alloy composition, phase, and property data from literature will be accomplished via text-based processing techniques such as regular expression (regex) matching and ML-based NLP. An overview of the process is as follows: from published experimental literature, we extract data records in a structured format. We then use data visualization techniques on those records to analyze and validate the data. Details of our methods are expounded below.

4.2.1 Data sources

The process begins with article retrieval to compile an article database. Unlike the previous chapter which only analyzed abstracts, here we need the full body text of articles, including tables, captions, etc. Fortunately, we are able to heavily leverage the existing infrastructure developed by our group over the past number of years. Starting from the CrossRef API to compile lists of DOIs, we use various methods to access and download the full text of articles, primarily in HTML or XML format. Our pipeline then parses, classifies paragraphs, and extracts synthesis information from the papers, storing all of it in JSON format in a MongoDB collection.

Our main article database contains the full text of over 4 million journal articles as of writing, from the following publishers, with whom our group has negotiated licensing agreements: [publishers]. Complementary to this is the extraction and parsing of tables from articles, and the associated table database. This is accomplished with the table extractor our group has developed, details of which can be found in [cite]. The table extractor was run on about 3.6 million manuscripts, the size of our article database at the time. The extracted table data is also stored in JSON format in a separate MongoDB collection as the articles, along with the source DOI.

From here, we face the crucial task of down-selection, that is, finding all of the papers and tables relevant to aluminum alloys. 140k Scopus about Al, 95k with DOI, 36k with full text

1. 222k: TITLE-ABS-KEY(aluminum AND alloy): “Broadly on-topic”, ok when

doing super specific body text search (for compositions)

2. This is a refinement of the query used to generate the “broadly on-topic” set from the previous chapter, which was found to include too many irrelevant papers that merely mention aluminum or alloy in passing. 153 k with DOIs 36 k with full text (in 3.6 M)

List of Sources: manuscripts from 77 journals

3. Full text comps from 5,172 unique DOIs: seems low?

4. Table comps Al balance ONLY, from 2,882 unique: seems right

5. Table props strength AND MPa, from 349 unique, associate series

a. More details about selecting relevant (e.g., excluding papers comparing with steel alloy)

4.2.2 Dataset preparation

Table extraction: finding and extracting tables from documents. [Zach]

Table information extraction: extracting information in structured manner from the tables. This is more complex task than table extraction, as table extraction is only the first step, while understanding the roles of the cells, rows, columns, linking the information inside the table and understanding the information presented in the table are additional tasks necessary for table information extraction. [11][12][13]

Composition and properties

1. Body text comps: Al-##Mg-, wt% unless says at%

a. DOIs kept for duplicates

2. AA comps: avg range, 0 as lower if max value given, 500 entries

3. Table comps: Al balance, convert at% when known

a. Originally also >50%, >85%, etc.

4. Patent comps: check Al in title, also from tables

Standard naming conventions, match formatting patterns for comp

1. 342 with UTS >1000 MPa removed

2. YS >400 MPa checked, some formed via severe plastic deformation, flagged

3. Elongation >50% manually checked for 5, 6, 7000 series, 45 removed (welds)

4. Technical validation: YS from Ansys Granta Edupack, outside range checked

5. Regex for temper designation

Phase extraction will be done through regular expression matching.

4.3 Results and Discussion

4.3.1 Composition data

1. 14,884 total alloy comps (Table 2 description of attributes)

2. Sources: full text (4,958), table (5,227), named (550), patents (4,149)

3. 69 elements, wt%, duplicates within and between source types kept

4. Name: pub # (patents), AA des (named), row name (table)

a. Emphasis on usefulness of extraction designation from row names/captions when available

5. AA_des from row name or caption, should add back series + explanation: from A_des, then primary (0.5-10 Si:Mg 6000)

6. ft_doi_list and table_doi

iv. Phase data?

4.3.2 Properties data

1. 1,278 mech prop entries (some with more than one prop)

2. DOI, (row) name, caption, AA_des when available

3. Series: from AA_des, else abstract (DOI series dict), else compositions (see above)

4. YS, UTS, elong, temper (O, F, W, H#, T#), flag, flag_note

iii. 2 main levers: composition (bigger role), processing

1. 2000, 6000, 7000 Heat treatment precipitation strengthening

2. 3000, 5000 mainly solution strengthening, strain hardening

3. Zinc highest solubility (7000)

4.3.3 Data visualization

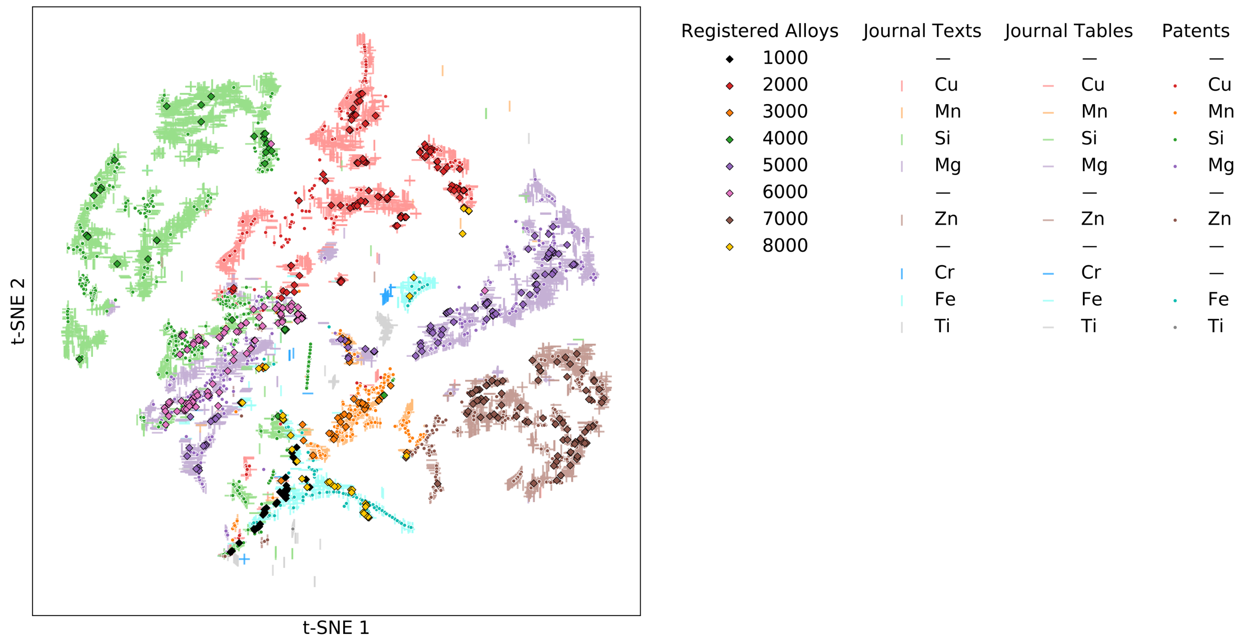


Figure 1: Overview of how an embedding is generated for a target node A, by aggregating the messages received from its neighbors, who in turn receive from theirs, to an arbitrary depth. This is a “depth-2” example. After [22]

1. 2, 3, 4, 5, 7 align with Cu, Mn, Si, Mg, Zn

2. 1000 various “alloying” = impurity

3. 6000 overlaps both Si and Mg

4. 8000 mostly Fe (what about Li?)

5. Small clusters, Explore wider space

vi. Swarm plot for YS

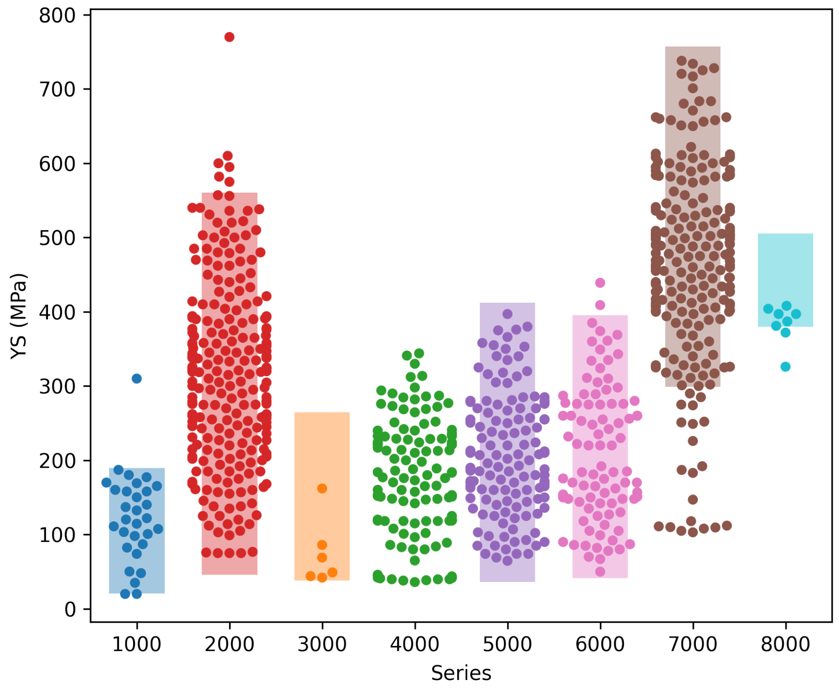


Figure 2: Overview of how an embedding is generated for a target node A, by aggregating the messages received from its neighbors, who in turn receive from theirs, to an arbitrary depth. This is a “depth-2” example. After [22]

1. 1000 lowest, 8000 highest, no range for 4000

vii. Scatter plot elong vs YS

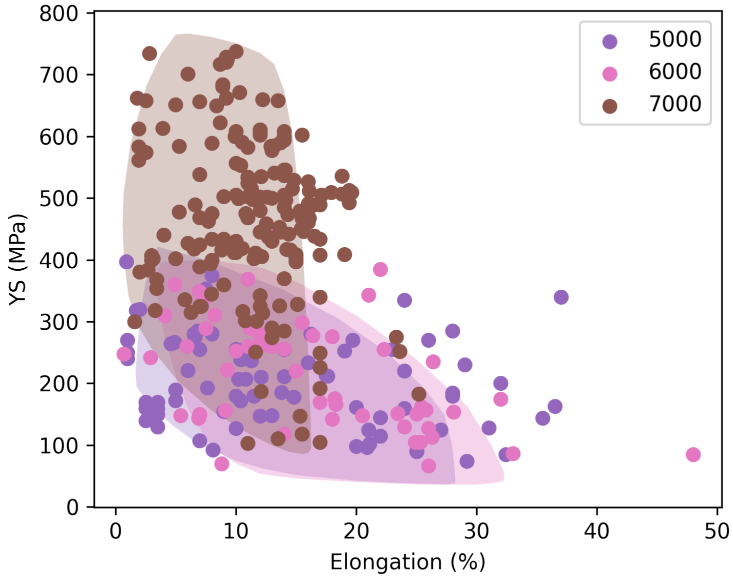


Figure 3: Overview of how an embedding is generated for a target node A, by aggregating the messages received from its neighbors, who in turn receive from theirs, to an arbitrary depth. This is a "depth-2" example. After [22]

1. Matches well with Edupack

4.4 Future Work

i. Further data extraction

1. Patents

2. Casting vs wrought

a. Casting alloys are cheap

b. Sheet can be recycled, melted

3. Fracture toughness by unit

a. Strength vs toughness

b. Everything related to precipitates/grain size

4. More processing details from current extracted

a. Associating specific treatment/properties with specific registered alloy

ii. Sentiment analysis of phases (effect on properties)

1. Associating phases with their role as precipitates, etc.

2. Which phases degrade formability? – clue for recycle friendliness

3. Relationship between phases and properties

iii. Disambiguation (NLP task): given sample names can we get corresponding experimental procedure (processing) and properties

1. Tie back to sample names section

2. Match sample to property in paragraph text using regex (insufficient, too many variations) tried for Al alloy dataset

3. Tim: try to build a model to detect "experimental variables" e.g., looking at "varied temperatures (100 °C, 130 °C, 160 °C and 190 °C) for 3 h" and predicting that those will be aligned one-to-one with the sample names, and doing that alignment

a. Learn to predict with weak supervision?

Disambiguation (NLP task): given sample names can we get corresponding experimental procedure (processing) and properties

Tie back to sample names section

Match sample to property in paragraph text using regex (insufficient, too many variations) tried for Al alloy dataset

Tim: try to build a model to detect "experimental variables" e.g., looking at "varied temperatures (100 °C, 130 °C, 160 °C and 190 °C) for 3 h" and predicting that those will be aligned one-to-one with the sample names, and doing that alignment

Learn to predict with weak supervision?

5 Sample Name Recognition

In the previous chapter, we are not yet be able to connect yield strength, temper, measures of ductility, and composition at an experiment level, rather it was at a document level, linked by DOI.

Sample names connect composition, synthesis, processing conditions and properties, results at sample, experiment level

Identify samples based on table row names (Table extractor)

Compositions from text e.g., Al-0.5Cu-2Mg (fixed structure)

Regex another way of extracting sample name if very specific field

Both often are or serve as sample names

Our contributions in this section are as follows:

Applying BERT to the task of sample name identification in scientific literature for the first time, as a first pass without significant modifications. The achieved F1 scores can serve as a baseline for more advanced models.

Performing hyperparameter grid search over optimizer learning rate and gamma parameter in focal loss function to determine best performance.

Interpreting the black-box classifier by looking at which words in a token's context contribute most significantly to its label prediction.

5.1 Task Definition

In any given materials science (or any scientific) paper, there will usually be numerous experiments or samples being studied, which might differ in composition, processing, etc. These would be referred to throughout the text by some sort of shorthand or other unique identifier, which we call a sample name (synonymous with experiment name in [1]). One can imagine then, that these sample names are what link, for instance, the synthesis conditions in the experimental section with measured properties in the results. Identifying these links and looking for patterns therein would greatly accelerate the design of new materials.

At a high-level, these links are spread throughout a paper and highly non-local. As a result, we expect the incorporation of document-level context to be important in accurate extraction of sample names, since it would be difficult even for humans to identify them based on just one sentence without context. To this end, we employ BERT (Bidirectional Encoder Representations from Transformers), the latest in contextualized word embeddings. BERT is an attention-based model, which is believed to mimic or better reflect how humans use context to infer meaning while reading. Regina non-local paper

Relation to named entity recognition: Sample name identification is very similar in implementation to the well-studied NLP task of named entity recognition (NER), namely that they are both token-level classification tasks. The CoNLL-2003 dataset [8] is a standard benchmark in NER, and one of the tasks BERT was evaluated on by Devlin et al. [2]. In one sense, NER is similar to sample name identification in that named entities are usually proper nouns, and many will be out of vocabulary, especially foreign or uncommon names. However, sample name recognition is also quite a different task from NER. Named entities are generally not unique to a single document, whereas most sample names are so unique as to only appear in a single paper. This suggests different training methods might be required as compared to NER.

Relation to new annotation scheme: Material, Target, Unspecified-Material, Nonrecipe-Material, **Sample**; Meta, Operation, Nonrecipe-Operation; Numbers, Units, Apparatus

Translate from / correspondences with NLP literature on Facet-disentangled Sentence Similarity (CoreSC); check emails, meeting notes

5.2 Methods

5.2.1 Dataset preparation

The dataset was compiled and released by the Authors of [1]. It consists of a list of 13,827 papers together with the sample names used in each. Examples of sample names can be seen in Table 1.

Table 1

To obtain a sufficiently large training set, an automated approach was used to extract the sample names, rather than fully manual labeling. In short, names found in tables under headers such as "experiment names" or "specimens", and appear elsewhere in the main text, were taken to be true sample names. Manual verification on 50 papers found an F1 score of 92.4. with precision higher than recall. Further details on the dataset can be found in [1].

An implicit but important assumption we are making is that papers which have samples names in tables talk about them in the main text in fundamentally the same way as papers that only have them in the main text. To confirm this hypothesis, we'd have use our trained model to predict on random papers (not just ones with sample names in tables), and manually verify any sample names identified.

The above dataset as released was in the form of a list of DOIs as strings, each with a sub-list containing its paper's associated sample names as strings. The papers were split 70- 15-15, so that there were 9611 for training, 2107 for vali- dation, and 2109 for testing. The actual content of the articles is stored in our database of over 2.5 million papers in a standard structured format. Each paper was automatically parsed into separate paragraphs, and the text from each paragraph was saved as plaintext strings. These data needed several steps of preprocessing to prepare it as input to BERT.

The first step was to actually label all the sentences with the ground truth of whether each token is a sample name. To do this, we used spaCy to sentencize and tokenize the raw paragraph text. Because we don't need or want any de- pendency parsing or automatic NER, a "blank" English language model was loaded. This meant there was no dependency parsing-based sentencization, only rule-based. After noticing that it didn't split on periods in certain cases, we added a rule to fix the run-on sentences this caused. In the future, we may want an even simpler sentencizer, with just one manual rule, to split only on the sequence period+ whitespace. We also didn't want NER because the only labeled entities we care about are the sample names. This was done using the EntityRuler, which can match each paper's sample name strings exactly (taking the longest span in case of overlap), and apply the label SNAME to match- ing tokens. These labels (or lack thereof) were then con- verted into the BIO-schema (denoting a token being at the Beginning, Inside, or Outside of an entity), which has become standard in NER.

A key feature of BERT is using WordPiece tokenization, which handles out-of-vocabulary words by breaking them down into smaller in-vocabulary pieces. Some have claimed this is as important an advancement as BERT itself. We use the Cased version of BERT Base (with its pre-trained model w eights checkpoint, config, and vocab), which has been shown to perform better on NER tasks than the Uncased version. This makes intuitive sense for our task, since sample names often use capitalized abbreviations. BERT Large was not used due to the low marginal benefit to computational cost ratio. So, per the BERT readme, the next step was to retokenize the labeled spaCy tokens, keeping a map- ping between token positions, so that a one-to-one correspondence between labels is maintained. Three datasets of varying sizes were thus processed, with different constraints imposed, as summarized in Table 2. The idea was the small set would be used for an initial test run, as well as repeated training when tuning hyperparameters. The paragraph cut- off length of 20 tokens for the large set was chosen because the vast majority of, if not all, paragraphs shorter than that are things like references and affiliations, which we don't care about. Moreover, the distribution of paragraph lengths falls sharply past 20, so this reduced the number of sentences to process by about 50%, significantly cutting the time taken.

We note that this retokenization step takes way longer than expected, seeing as it's just a deterministic lookup process and not actually doing anything "meaningful". This is probably because it's retokenizing one token at a time, and only using the CPU. The only reason we had to do it this way, rather than tokenize with BERT directly, was to be able to first match and label the tokens using spaCy, since BERT doesn't come with a string matcher. For sentence- level tasks (e.g. paragraph classification), we can just tokenize entire sentences directly into BERT, which should be faster. Another way would be to look into a new release of spaCy that comes with BERT models, so that entity matching and tokenization into BERT WordPieces can happen in one step.

The last few steps were to use the mapping to label the BERT tokens, label all non-initial WordPiece tokens (which start with ##) as X, pad or trim all sequences to 128 tokens (since model inputs must have a uniform shape), and generate masks to ignore the predictions on all padding tokens. The sequence length of 128 was chosen because that included over 97% of sentences in the small set (vs. 99.6% with 256 and 99.95% with 512, the maximum BERT supports), and it was a good comprise between input length and batch size, which affects training time. An important point to note is that Keras automatically masks padding in inputs (with mask_zero=True), and propagates this mask all the way through to the loss function. Our explicit masks are still needed to trim predictions before generating confusion matrices and metrics with the PyCM package.

5.2.2 Model architectures

tf–idf We note that experiment names tend to be used relatively frequently within a document, but that across documents a specific experiment name tends to be rare or unique. Formalizing this idea, we consider the term frequency $tf_{t,d}$ of term t in document d to be the count of t in d divided by the total number of terms in d. Similarly, we take the document frequency df_t

to be fraction of documents that contain term t. One might expect that for $t \in S_d$ where S_d is the set of experiment names of document d, $tf_{t,d}$ is high but df_t is low. It is then reasonable to expect that the term frequency–inverse document frequency metric (tf–idf), as given in Equation 1 could be used to identify experiment names (Manning et al., 2008). We therefore use this as a baseline "zero" model.

Reasoning over every token in a document is computationally intractable if we need to draw in information from other parts of a document for each token. Our strategy is therefore to first propose a list of candidate experiment names, and then filter this list by incorporating further information for each candidate of the list. The model is composed of two stages. A schematic is provided in Figure 1.

Figure 1: One- and two-stage models

Baseline one-stage model: Candidate Generator Model 1 takes every sentence in a document, and then for each token in that sentence learns the probability that it is an experiment name. In other words, it models Pr(ti,s,d ∈ Sd|ts,d), where ti,s,d is token *i* in sentence s of document d, Sd is the set of experiment names of document d and ts,d is the sequence of tokens that constitute sentence s in document d.

There are in general two ways of using BERT as mentioned in [2]: fine-tuning and a feature-based approach. In brief, the fine-tuning approach takes the entire BERT model and adds one or two final layers on top for the specific task, then trains all the weights in the combined model to perform that task. The feature-based approach on the other hand is more akin to classic word embedding models like ELMo or word2vec, where fixed feature vectors (aka embeddings) are extracted from the model then fed as input into neural nets of any architecture to perform the downstream task. In this work we used fine-tuning, because most prior work by our group and others used the feature-based approach, and we wanted to explore this new paradigm.

Models were implemented with Keras to maintain compatibility with our existing pipeline infrastructure, in consideration of its potential future BERT integration. We take the output of the last hidden layer in the basic BERT model, of shape (batch size, 128, 768), pass it into a 0.1 dropout layer, followed by a dense linear layer with a softmax activation to four outputs, one for each of the labels B, I, O and X, so that the final output is of shape (batch size, 128, 4). The output with the highest probability is taken to be the model's prediction on a token.

Model 1 is tuned to have high recall at the cost of lower precision. Explicitly increase recall by lowering threshold below 0.25 (BIOX) or 0.5 (binary) – right now precision, recall similar

Two-stage model: Candidate Generator plus Filter We append a Model 2 to the output of Model 1, that serves to filter the high-recall list of candidate experiment names identified by Model 1 by incorporating document-level context. Here we model the probability that candidate cj is an experiment name, given the context for this candidate: Pr(cj,d ∈ Sd|$context_j$). Context is incorporated from select sentences and from the strings of other candidates. Concretely, for each candidate, k sentences containing that candidate selected at ran- dom. These sentences are then processed using an LSTM. The outputs of each of these LSTMs are summed to provide a permutation-invariant cumulative signal with context from the entire document. Further, we find that physical experiments often follow an incremental naming scheme (e.g. *sample A*, *sample B*, *sample C*). To capture this structure, we also include the Levenshtein (Levenshtein, 1965) edit distance between each candidate and all other candidates in our model; experiment names tend to have low edit-distances to all other experiment names, but high edit-distances to everything else. Finally, the candidate itself is embedded. The signal from the LSTMs, the Levenshtein distance, and candidate embedding are con- catenated and provided to a dense neural network before passing through a sigmoid to make a final binary prediction, as in Model 1.

Loss function: We first tried using categorical cross-entropy loss to train with accuracy as the metric, which is typical for multi-class classification. We soon realized a major flaw: because the majority classes (O and X, not sample name) outweigh the minority class (B and I, sample name) roughly 100 to 1, the model can just guess "not sample name" for everything and still get 0.99 accuracy. This needs to be dealt with while training, and is also why F1 is the actual metric we should use. The usual way to deal with a class imbalance is using class weights, by making the majority class contribute less in fitting. But this doesn't work directly with time-series or sequential data, like sentences.

Two different solutions were tried. First, we used sample weights, which allows each individual sample to be weighted differently, and in temporal mode, which further allows each "timestamp" (i.e. token) in each sample to have a different weight. The weight of each token was set to 1/(fraction of all labels its label's class accounts for). This weighs each label's contribution to the loss function inversely to its class's frequency. The second was a more elegant, robust and computationally cheaper solution: using the focal loss function introduced by Lin et al. [9]. Focal loss, so named because it focuses on the hard to learn examples, is defined for multi-class sequential data as follows:

$$FL(y,y)$$

where n is the sequence length, m is the number of classes, y is the one-hot encoded vector of true labels, y^ is the predicted softmax probabilities of every class for every token, and $|| \cdot ||1$ denotes the L1 norm. α and γ are balancing and focusing parameters respectively. Focal loss can be thought of as a weighted version of cross-entropy loss. Intuitively, the more easily a class is learned, the more confident the predictions for that class will be, so the higher yc is, $(1 - yc)$ means the less it will contribute to the loss. The normalization in Eq. 2 ensures that the weighing term is $\in (0, 1)$, and γ controls how much of a focusing effect this term has, the effect of which we also investigate.

5.3 Experiments and Discussion

5.3.1 Training

In both approaches mentioned earlier, the training is supervised. We initially used a batch size of 32 with one 12 GB GPU, as recommended in the BERT readme for sequence length of 128. According to the Keras documentation, using multiple GPUs speeds up training "quasi-linearly". So, we used Keras's built-in method to create a parallel model on our two GPUs to increase batch size on each step, and found in practice a maximum size of 56, as larger batches cause resource exhaustion (out-of-memory) errors.

For our optimizer, we used Adam with warmup, which helps prevent early overfitting. Adam [10] is an adaptive learning rate optimization algorithm, and one of the best performing and most commonly used ones. The nominal learning rate for the optimizer is the other hyperparameter we sweep over to find the optimum, especially as early experiments indicated a high degree of overfitting.

Custom metrics: Beyond the masking of padding tokens, we also want to ignore non-initial WordPiece tokens, and only count the label of a word's first piece as its prediction. This can be done in one step by taking the ground truth labels and seeing which indices are labeled 3 (i.e. X), and using this as a filter so that only true and predicted labels of actual words are compared. Precision, recall, and F1 were implemented as custom metrics in Keras, including the filtering step above. We couldn't simply use the implementation from sklearn metrics, since they don't handle multidimensional tf tensors natively, and would require reshaping and converting to use as metrics during training and evaluation in Keras. A point to note is that in Keras, these metrics are calculated (summed) per batch, and the values displayed while training is a running average, which, while generally close, is not the same as the final scores over the entire dataset. This was partly why we used the PyCM package to calculate metrics.

Hyperparameter search: For the first run, intended as a trial run, we made a 90-10 train-test split on the small dataset, then used 10% of the training set to validate. We trained for 7 epochs (approx. 4 hours), when the early stopping condition of increasing val loss over 2 epochs was met, and used the default learning rate of 10^{-4} and focal loss γ parameter of 2. With this, a validation F1 of 0.9982 and test F1 of 0.9408 was achieved, which was far higher than expected. The main reason this result was dubious was the complete lack of incorporating document-level context. We considered the possibility that a machine could learn to recognize more subtle patterns, in single sentences out of context, that can discern sample names in a way humans cannot. Though this seemed unlikely, it was the initial hypothesis explaining the high F1 score before we realized the real reason.

The actual problem was more subtle: when we made the train-test split, the data was shuffled before being split. Because the same sample names occur in multiple sentences, this meant that different sentences with the same sample names would randomly end up in both train and test sets. Therefore, the machine wasn't actually learning to recognize patterns in where sample names occur, rather, it was merely learning the correct labels for the specific token sequences for sample names that occur in this dataset. In other words, it was strongly overfitting. The correct way should have been to disable shuffling before splitting into train-validate-test, so that the split occurs at a document, not sentence, level. We also decided to split 70-15-15, to match the large dataset.

After the initial trail run's overly promising results, we actually went directly to the large dataset. This was when we first real- ized there was a problem, since the validation loss and F1 never improved, and in fact val loss only increased even as loss was decreasing. This generally indicates overfit- ting; the problem was it seemed to be overfitting right from the beginning. This was why a hyperparameter search was carried out, to test smaller learning rates.

With the correctly split train and val sets, we performed a hyperparameter grid search over learning rate lr = (10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}) and γ = (0.0, 2.0, 4.0), training for 5 epochs with no early stopping.

5.3.2 Results

The test F1 scores for hyperparameter search are shown in Table 3. Note that a γ of 0 is equal to using categorical cross-entropy loss, and the result below show that using focal loss does generally make an improvement. These results can also be compared with the initial (pre-hyperparameter search) result on the large dataset. We trained for 7 epochs (approx. 2 days) with the original lr = 10^{-4} and $\gamma = 2$, and no early stopping. The best F1 achieved was 0.5136, marginally better than that achieved on the small dataset with the same hyperparameters, at 0.4770.

Table 2: F1 results of hyperparameter grid search

These results don't tell the whole story however, since smaller learning rates naturally take longer to converge. Since our motivation for doing the grid search was to solve the problem of early overfitting (as seen in Figures 2a and 2b), it's important to look at the training curves. Loss and F1 over time for lr = 10^{-7}, $\gamma = 4.0$ are shown in Figures 2c and 2d, respectively. Crucially, the val loss and F1 followed the same trend as train loss and F1, as expected in proper training. It is clear that convergence has not been reached, due to the low test F1 score, but these experiments suggest a learning rate between 10^{-7} and 10^{-6} should be used, with $\gamma = 2.0$ or 4.0.

Figure 2: Training curves for various hyperparameter setting combinations.

A point to note is that the metrics in Keras are calculated (summed) per batch, and the values displayed while training or evaluation is a running average, which, while generally close, is not the same as the final scores over the entire dataset. This was why we used the PyCM package to calculate metrics, which further allowed us to see a wide variety of metrics, both overall, like MCC, and for each class. As expected, the F1 for classes 2 and 3 (O and X) were very high (close to 1) while that of classes 0 and 1 (B and I) hovered around 0.5 after convergence.

tf–idf 0.144 F1

One-stage model FT-w2v (Alex) 0.646 F1

BERT (6.883) 0.514 F1

SciBERT (Email to Emma) 0.667 F1

Two-stage model FT-w2v 0.704 F1

BERT, SciBERT, etc. TBD

Table 3: Summary of Results

This F1 score is rather mediocre. In fact, it's lower than that achieved by Model 1 alone (F1 = 0.646) in [1], which was a simple BiLSTM GRU architecture also taking in single sentences. We had originally expected BERT to do better than this model, but worse than the two-stage Model 2 (F1 = 0.704). It's not clear whether this is down to the classifier architecture (in this work, we used a simple dense linear layer after the Transformers).

When we first got the high F1 values on the small dataset, we extracted the tagged sample names to compare with the manually labelled original sample names, primary as a check that the tokens were actually labeled correctly. Generally speaking, there was a very good match, which, in retrospect after realizing the shuffling issue, shouldn't be surprising.

5.3.3 Explainable classifier

As part of our on-going goal to create interpretable models, we tried to gain some insight into the black-box classifier using the LIME algorithm [11], implemented as part of ELI5, which works for explaining any classifier. Intuitively speaking, LIME works by learning an approximate white-box classifier, by perturbing the input to see which features or tokens have the greatest effect on the prediction. Here, we were interested in seeing what sort of patterns the model learned that was seemingly able to identify sample names so well. On a technical note, we had to modify the source code of the ELI5 package to allow it to accept pre-tokenized text as input. Otherwise, the package accepts strings of text by default and splits purely on whitespace, which wouldn't align with our labels.

Figure 3 shows a random sampling of LIME results, representative of a few types of predictions, both correct and incorrect, with some discussion in the captions. The conclusions are probably less valid having realized the overfitting issue, nevertheless some of the explanations are plausible and interesting.

Figure 2: Sampling of LIME token classifier explanations

Overall however, these explanations must be taken with a grain of salt, as there is likely a lot of confirmation bias, since this was before we realized the model was probably just learning the sample names themselves due to the shuffled split. SHAP values

5.4 Future Work

Clearly, there is still much work to be done on the task of sample name identification. The most obvious next step would be to combine the present work with that of [1], that is, to use BERT with the feature-based approach. We would extract word embeddings as one or more hidden states of BERT, then feed into the one- and two-stage models. Currently, the Authors use FastText embeddings pretrained on a materials science corpus. We could do the same thing with BERT, using the two unsupervised pretraining tasks from [2], Masked Language Modeling and Next Sentence Prediction. The hope is that this provides better language understanding in general, with learning of context implicitly embedded in the model, to improve performance on specific downstream supervised tasks.

We hypothesize that a major reason for the poor performance of our model is the lack of explicitly incorporating any document- or even paragraph-level context. We only input single sentences of up to 128 tokens. We could try longer input sequence, but that captures paragraph-level context at best, and also needs much more memory. Adding context through a smarter approach (sampling sentences containing a possible sample name) was a key feature of the two-stage model developed in [1].

We thought of a few other variations on the idea of in- putting multiple sentences (BERT supports up to 512 tokens input all at once). One idea was to "pad" each sentence with the ones immediately before and after. Thus, there would be continuous overlap in the inputs, and we wondered if this could somehow capture document-level context. Of course, the downside is that it would be three times slower. A simi- lar idea is to input 5 sentences at a time, but only overlap the front and back ones, so that 3/5 of the sentences will only get input once. In either approach, we would collect all to- kens predicted to be a sample name on at least one instance. Then we'd get all instances of that word's predictions, and say it's a sample name if it's predicted to be one in more than half (or some higher threshold) of instances.

Another problem was that WordPiece might break up sample names and technical terms too much, since they're so specialized and out-of-vocabulary. Since they get tokenized into so many small pieces, as seen in Figure 3c, little meaning remains and it likely can't make use of learned contextualized embeddings from pretraining. To address this, we could see if SciBERT [5] makes a difference. At least its vocabulary includes more scientific terms than BERT's base vocabulary, which could mitigate the breaking up of certain words.

Sample/weigh based on prediction threshold

Possible (stage 1.5) identify relevant sentences by end-to-end score ranking (need to

clarify with Tim)

Include sentences with OTHER close (Levenshtein or cosine) sample names e.g., H400

for H600

Smaller lr, more epochs (compare with SciBERT rate? Loss function?)

6 Outlook and Conclusions

While machine learning and materials informatics have produced many useful results over the past several years, there are still many challenges to overcome, such as scarcity and sparsity of data. Incorporating domain knowledge of any form, such as phase diagrams or the physics governing kinetics, into ML models should produce much better results and an interpretable model could even lead to the discovery of new physical laws.

Challenges

With NLP for materials:

- Transferability across materials domains
- Solid state, thin film, templated synthesis
- Off-stoichiometry “BaxMn1-xO3 for x = 0.9”
- Lexical ambiguity and evolution

With overall approach:

- Age, quality Optical character recognition, pdf vs. html
- Linked recipes “following the method described by....”
- Negative examples Bias of literature toward success

Example from chemical synthesis

- Data augmentation reaction databased supplemented with chemically plausible negative examples ACS central science, 2017

NLP activities

- Entity extraction, Event extraction, Relation extraction, Entity linking
- Chemical Entity Recognition
- Parts of a text, types of texts

Concluding thoughts on NLP progress

- Natural language processing is young in its application to materials science
- It takes effort to build up an annotation approach and corpus
- There are domain-specific needs regarding accuracy and ambiguity
- Tradeoff between accuracy and degree of generalizability

Bibliography

[1] "Materials Algorithms Project." https://www.phase-trans.msm.cam.ac.uk/map/mapmain.html (accessed Jul. 16, 2019).

[2] B. D. Conduit *et al.*, "Probabilistic neural network identification of an alloy for direct laser deposition," *Mater. Des.*, vol. 168, p. 107644, Apr. 2019, doi: 10.1016/j.matdes.2019.107644.

[3] T. F. G. G. Cova and A. A. C. C. Pais, "Deep Learning for Deep Chemistry: Optimizing the Prediction of Chemical Patterns," *Front. Chem.*, vol. 7, p. 809, Nov. 2019, doi: 10.3389/fchem.2019.00809.

[4] S. Wang *et al.*, "Massive computational acceleration by using neural networks to emulate mechanism-based biological models," *Nat. Commun.*, vol. 10, no. 1, p. 4354, Dec. 2019, doi: 10.1038/s41467-019-12342-y.

[5] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 358, p. 112623, Jan. 2020, doi: 10.1016/j.cma.2019.112623.

[6] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, no. 1, pp. 477–508, Jan. 2020, doi: 10.1146/annurev-fluid-010719-060214.

[7] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, "Machine learning in materials informatics: recent applications and prospects," *Npj Comput. Mater.*, vol. 3, no. 1, p. 54, Dec. 2017, doi: 10.1038/s41524-017-0056-5.

[8] G. Carleo *et al.*, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, no. 4, p. 045002, Dec. 2019, doi: 10.1103/RevModPhys.91.045002.

[9] B. Meredig, "Five High-Impact Research Areas in Machine Learning for Materials Science," *Chem. Mater.*, vol. 31, no. 23, pp. 9579–9581, Dec. 2019, doi: 10.1021/acs.chemmater.9b04078.

[10] J. Hill, G. Mulholland, K. Persson, R. Seshadri, C. Wolverton, and B. Meredig, "Materials science with large-scale data and informatics: Unlocking new opportunities," *MRS Bull.*, vol. 41, p. 11, May 2016, doi: 10.1557/mrs.2016.93.

[11] G. R. Schleder, A. C. M. Padilha, C. M. Acosta, M. Costa, and A. Fazzio, "From DFT to machine learning: recent approaches to materials science–a review," *J. Phys. Mater.*, vol. 2, no. 3, p. 032001, May 2019, doi: 10.1088/2515-7639/ab084b.

[12] D. Jufrasky and J. H. Martin, "Information Extraction," in *Speech and Language Processing*, 3rd ed. draft., 2021. Accessed: Dec. 14, 2021. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/17.pdf

[13] N. Sager, *Natural Language Information Processing: A Computer Grammmar of English and Its Applications*. Addison-Wesley Longman Publishing Co., Inc., 1981.

[14] J. Cowie and W. Lehnert, "Information extraction," *Commun. ACM*, vol. 39, no. 1, pp. 80–91, Jan. 1996, doi: 10.1145/234173.234209.

[15] M. T. Pazienza, *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97 Frascati, Italy, July 14-18, 1997*. Berlin, Heidelberg: Springer-Verlag Springer e-books, 1997.

[16] Y. Wilks, K. Ahmad, C. Brewster, and M. Stevenson, *Words and intelligence*. Dordrecht: Springer, 2007.

[17] G. DeJong, "An Overview of the FRUMP System," in *Strategies for Natural Language Processing*, vol. 113, 1982, pp. 149–176.

[18] J. R. Cowie, "Automatic analysis of descriptive texts," in *Proceedings of the First Conference on Applied Natural Language Processing*, 1983, p. 7.

[19] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic Keyword Extraction from Individual Documents," in *Text Mining*, John Wiley & Sons, Ltd, 2010, pp. 1–20. doi: 10.1002/9780470689646.ch1.

[20] F. Chollet and others, "Keras," 2015, [Online]. Available: https://keras.io

[21] Z. Jensen *et al.*, "A Machine Learning Approach to Zeolite Synthesis Enabled by Automatic Literature Data Extraction," *ACS Cent. Sci.*, p. acscentsci.9b00193, Apr. 2019, doi: 10.1021/acscentsci.9b00193.

[22] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," *ArXiv170905584 Cs*, Apr. 2018, Accessed: Jan. 12, 2020. [Online]. Available: http://arxiv.org/abs/1709.05584