

From Descriptive to Predictive: Forecasting Emerging Research Areas in Software Traceability Using NLP from Systematic Studies

Zaki Pauzi^a and Andrea Capiluppi^b

Bernoulli Institute, University of Groningen, The Netherlands

Keywords: Natural Language Processing, Software Traceability, Time Series, Systematic Review.

Abstract: Systematic literature reviews (SLRs) and systematic mapping studies (SMSs) are common studies in any discipline to describe and classify past works, and to inform a research field of potential new areas of investigation. This last task is typically achieved by observing gaps in past works, and hinting at the possibility of future research in those gaps. Using an NLP-driven methodology, this paper proposes a meta-analysis to extend current systematic methodologies of literature reviews and mapping studies. Our work leverages a Word2Vec model, pre-trained in the software engineering domain, and is combined with a time series analysis. Our aim is to forecast future trajectories of research outlined in systematic studies, rather than just describing them. Using the same dataset from our own previous mapping study, we were able to go beyond descriptively analysing the data that we gathered, or to barely ‘guess’ future directions. In this paper, we show how recent advancements in the field of our SMS, and the use of time series, enabled us to forecast future trends in the same field. Our proposed methodology sets a precedent for exploring the potential of language models coupled with time series in the context of systematically reviewing the literature.


1 INTRODUCTION


In the field of software engineering, systematic reviews have gained considerable popularity since the introduction of Evidence-Based Software Engineering (EBSE) guidelines in 2004 (Kitchenham et al., 2004); a framework to assess the quality of primary studies through aggregating all empirical studies on a particular topic. A simple search in Google Scholar for “systematic reviews in software engineering” showed a growth trend over the past years, with 2021 results returning 300% more than in 2012¹.

Despite the increasing popularity behind systematic reviews in software engineering, a key challenge in conducting these still remains: balancing between methodological rigour and required effort (Zhang and Ali Babar, 2013). This paper addresses this challenge by leveraging recent language models and time series

analysis to conduct a predictive analysis of trends in research focus. The approach that we show has the potential of reducing the manual effort required by researchers, yet enhancing the methodological rigour in reviewing literature. Based on the lessons outlined from applying the systematic reviewing process (Brereton et al., 2007), our proposed methodology is a result of the following:

- *L15: Software engineering systematic reviews are likely to be qualitative in nature.* Using NLP tools and techniques supplements this reviewing process in providing a predictive analysis to answer the research questions at hand. Descriptive qualitative analysis may not be enough to fully answer the research questions, and recommendations typically are based on manual analysis and interpretation.
- *L18: Review teams need to keep a detailed record of decisions made throughout the review process.* Transparency in the decision-making process improves explainability and supports effective reviewing process. Our proposed methodology involves NLP models such as word embeddings that are agnostic to individual perceptions (understanding) of semantics but rather representative of

^a  <https://orcid.org/0000-0003-4032-4766>

^b  <https://orcid.org/0000-0001-9469-6050>

¹This search was done on incognito mode filtering ranges of each year and only the review articles. An example URL for the year 2021: https://scholar.google.com/scholar?q=systematic+reviews+in+software+engineering&hl=en&as_sdt=0%2C5&as_rr=1&as_vis=1&as_ylo=2021&as_yhi=2021

(by training) large amounts of corpus text from multiple sources.

The work that we present in this paper is based on the data gathered from our own systematic mapping study (Pauzi and Capiluppi, 2023). In that review, we studied which trends have emerged in NLP techniques to support or automate the tasks of software traceability. The analysis was conducted within the software development life cycle (SDLC) framework, and various research trends emerged from the evaluation of past works.

Joining predictive analytics and our own systematic study (but potentially, any other systematic review), we observe how the importance of certain terms evolved over the past years; specifically for *requirement* and *design*. Both of these terms were chosen as they represent two core phases of the SDLC that constitute a major chunk of traceability efforts using NLP (Pauzi and Capiluppi, 2023). We also analyse the semantically *relevant* terms to these and forecast the importance of these two terms for the next two years.

2 RATIONALE AND RESEARCH QUESTIONS

2.1 Real-World Applications of Predictive Analytics

Identifying and forecasting emerging trends is a part of predictive analytics, enabling stakeholders (such as policymakers, executives, etc.) to make better-informed decisions. We borrow from some recent real-world applications and case studies to propose an innovative approach to use the results of a(ny) systematic literature review.

In the energy industry, predictive analytics is a particularly crucial component, due to global efforts to reduce greenhouse gas emissions by shifting energy sources to sustainable ones. Predictive analysis is done by mining text from patents, scientific publications, and Twitter data. Given the disruptive impact it has on the energy industry, forecasting emerging trend is undoubtedly critical for government R&D strategic planning, social investment, and enterprise practices. In the photovoltaic industry, social media (tweets) were mined to construct an evolution map of Twitter users' sense of and response to perovskite solar cell technology (Li et al., 2019).

For analysing patents, pyGrams² is an open-

²<https://github.com/datasciencecampus/pygrams>

source tool to discover emerging terminology in large text datasets. The quantitative review of emerging terminology from patent applications is an important objective for the Intellectual Property Office (IPO). A similar concept is demonstrated in this paper.

2.2 Research Questions

Given the applicability of predictive analytics in diverse industries, we aim to replicate these applications, but in the context of systematically reviewing the literature. We formulate the following research questions:

RQ1: How have specific terms emerging from a systematic review evolved in the past?

RQ2: What are the closest terms (in terms of semantic similarity) to these?

RQ3: What is the *forecast trajectory* for the importance of these specified terms?

3 BACKGROUND RESEARCH

Text mining on publications has been explored for a variety of reasons; mainly to automate the manual process of reviewing relevant literature (Thomas et al., 2011). Examples include an automated screening of studies for selection (O'Mara-Eves et al., 2015), improving search strategy (Ananiadou et al., 2009; Stansfield et al., 2017), automatically extracting an ontology of relevant topics (Osborne et al., 2019), and automating the selection strategy such as facilitating the identification, description or categorisation, and the summarisation of relevant literature (Thomas et al., 2011). Besides text mining as a part of the systematic process, analysis of the use of text mining in systematic reviews has also been done previously, such as for the citation screening stage (Olorisade et al., 2016).

Many challenges faced by systematic reviewers are similar across different disciplines, with software engineering being no different (Marshall et al., 2015). To support reviewers in this arduous quest, readily available tools were developed, providing text mining as part of its core capabilities, such as the Systematic Review Toolbox (Marshall and Brereton, 2015). By combining NLP and time series, we aim to achieve a more prescriptive-focused standpoint to expand the current systematic review process in software engineering. A similar work was based on ACL anthology (Asooja et al., 2016): although the premise was similar, the approach was specifically using only one conference proceedings without a specific topic in question.

4 METHODOLOGY

Figure 1 shows a simplified workflow diagram to represent the sequence of steps in our proposed methodology. The full notebook can be found online and used freely for further queries and results ³.

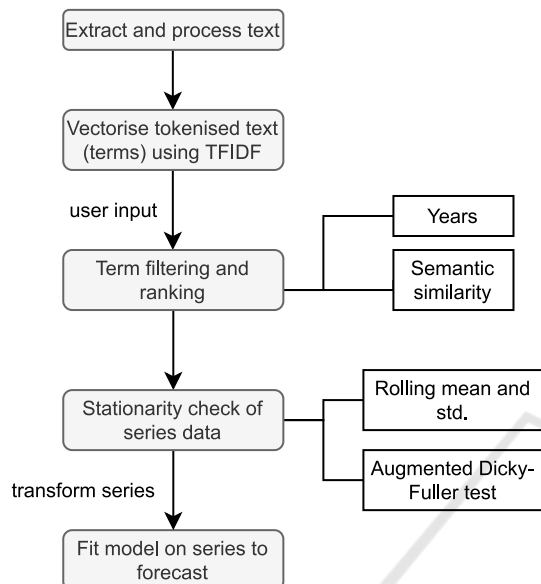


Figure 1: Sequence of steps taken in our methodology.

4.1 Dataset

Our dataset of publications is based on the selection criteria of a previous mapping study (Pauzi and Capiluppi, 2023): a total of 96 papers were obtained, all dealing with NLP techniques to support (or automate) software traceability. The complete list of papers selected can be found online ⁴, covering a period of years 2013 to 2021.

4.2 Text Processing

Text content from every paper in scope was processed through the pipeline shown in Figure 1. The cleaned text from every paper (document) is then tokenised as unigrams, bi-grams, and tri-grams. The steps taken were the following:

1. Convert words into tokens (tokenisation)
2. Remove noise: keep only alphanumeric characters
3. Convert all to lowercase

³<https://github.com/zakipauzi/enase2023/blob/main/enase2023.ipynb>

⁴<https://github.com/zakipauzi/enase2023/blob/main/papers.csv>

4. Remove stopwords (using a standard NLTK package with additional words identified)
5. Convert tokens to their lemma form (SpaCy lemmatizer ⁵)
6. Remove letterhead or front page matter

4.3 Terms Frequency - Inverse Document Frequency (TFIDF)

These N-grams ⁶ are vectorised using Terms Frequency - Inverse Document Frequency (TFIDF). TFIDF Vectorizer is a feature extraction technique for large text corpus offered by scikit-learn (Pedregosa et al., 2011). The number of times a term ⁷ occurs (term frequency, tf) in a given document is multiplied with the inverse document frequency idf : $tf(t, d) * idf(t)$. This technique enables a more effective term weighting scheme compared to the sole count of terms as those very frequent terms would shadow the frequencies of rarer yet more important terms. The popularity of this term-weighting technique is vast: a survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use TFIDF (Beel et al., 2016). In this paper, the TFIDF score calculated of a term t (N-gram) extracted from the dataset is denoted as “ $tfidf_t$ ”.

4.4 Filtering and Visualising

We provide a historical descriptive trend of $tfidf_t$ for top-ranked terms (by N-gram tokenisation) based on two viewpoints: years trend (RQ1) and semantic similarity (RQ2). Every paper has a published date, which is used to group the $tfidf_t$ of each term corresponding to the years it is extracted from. This is then plotted on a line graph to show how these top-ranked terms change in $tfidf_t$ across the years.

In the viewpoint of semantic similarity, we use a pre-trained language model in the software engineering domain (Efstathiou et al., 2018). We vectorise the corpus tokens (N-grams) using the Word2Vec (Mikolov et al., 2013) model trained on Stack Overflow vocabulary dataset. This technique allows us to represent words in a vector space, based on the model that has been trained with a vocabulary in the context of software engineering. This is necessary to disambiguate polysemous words and detect synonyms by interpreting them in the context of

⁵<https://spacy.io/api/lemmatizer>

⁶A collection of N adjacent tokens. This can be a uni-gram (N=1), bi-gram (N=2), and so on.

⁷These are the N-grams extracted. For example, machine learning is a bi-gram term.

software engineering. The terms are then ranked according to the similarity scores calculated using cosine similarity to the user input’s term: a “distance” metric, irrespective of orientation and magnitude – the lower the angle between the vectors, the higher the similarity.

4.5 Time Series Analysis for Emerging Terminologies

The core feature of our methodology is the transition from descriptive analysis (typically, what systematic reviews focus on) to predictive analysis (i.e., what importance will a term have in the future). Due to the timestamps of publications, we are able to derive time series data that reflects a variate *time* to the equation. In answering RQ3, we present the following key steps:

4.5.1 Grouping Datapoints

Firstly, we sort the papers according to the published date in ascending order. The *timestamp* of every paper becomes the index in our table. The $tfidf_t$ of the user input’s term is grouped according to the timestamp. Where there are multiple values of $tfidf_t$ within the same period (month), we calculate the average.

4.5.2 Check for Stationarity

Prior to fitting our model, we need to check for stationarity — property that the mean, variance and autocorrelation structure do not change over time. In other words, a flat-looking series, without trend, with constant variance over time. To check this, we run the rolling statistics test and the Augmented Dickey-Fuller (ADF) test (Bilgili, 1998). If stationarity is not confirmed, we have to transform the series to remove patterns of trend and seasonality⁸. In our paper, we use the `statsmodels` package offered in Python⁹.

4.5.3 Fitting the Model

We use the auto-regressive integrated moving average (ARIMA) model to fit the datapoints reflecting $tfidf_t$ of user input across the series. Due to the limitation of space, we will be only exploring ARIMA model using `auto_arima` from the `pmdarima` package¹⁰. `auto_arima` operates like a grid search, in that

⁸Some examples: log transforming, square root, proportional change, etc.

⁹<https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html>

¹⁰<https://pypi.org/project/pmdarima>

it tries various sets of p and q parameters¹¹, selecting the model that minimises the Akaike Information Criteria (AIC)¹².

5 RESULTS

Our pipeline is fit to any user input that is within the vocabulary of the Word2Vec model; however, for space limitations, we will focus here on the two core phases of the SDLC, i.e., requirement and design. User inputs “requirement” and “design” represent a user interested to know how these topic terms have played a part in using NLP for software traceability over the past years.

5.1 RQ1

We calculate the TFIDF mean scores (mean $tfidf_t$) of requirement and design throughout the years in scope. requirement and design are highly ranked unigrams (in terms of $tfidf_t$) from our dataset, with the following timeline changes of mean $tfidf_t$ over the years, shown in Figure 2:

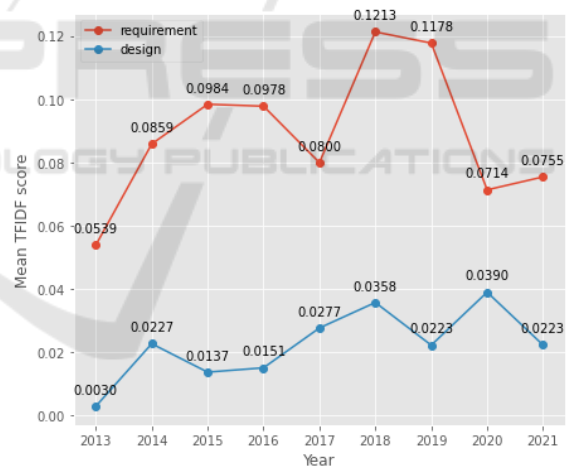


Figure 2: Mean $tfidf_t$ of requirement and design over the past years.

5.2 RQ2

Based on the user inputs, we also show the terms ranked in semantic similarity (dictionary filtering) with their respective sum and mean $tfidf_t$. Tables 1

¹¹ p is the number of autoregressive terms and q is the number of lagged forecast errors in the prediction equation.

¹²An estimator of prediction error which measures a statistical model in order to quantify the goodness of fit of the model.

and 2 show the top 10 terms for requirement and design respectively.

Table 1: Top 10 ranked by semantic similarity to requirement.

	Term	sum	mean	score
1	hipaa	0.4448	0.0046	0.4147
2	regulation	0.1010	0.0011	0.4070
3	constraint	0.1511	0.0016	0.3814
4	recommen- dation	0.1113	0.0012	0.3766
5	design	0.2311	0.0024	0.3412
6	feature	0.3203	0.0033	0.3404
7	concept	1.0397	0.0110	0.3363
8	architecture	0.4086	0.0043	0.3340
9	case	0.2604	0.0027	0.3218
10	knowledge	0.1159	0.0012	0.3102

Table 2: Top 10 ranked by semantic similarity to design.

	Term	sum	mean	score
1	architectural	0.3855	0.0040	0.6994
2	architecture	0.4086	0.0043	0.5899
3	modulari- zation	0.1881	0.0020	0.5181
4	approach	0.7675	0.0080	0.4818
5	composit- ional	0.1013	0.0011	0.4669
6	concept	1.0397	0.0108	0.4568
7	uml	0.1896	0.0020	0.4494
8	functional	0.1430	0.0015	0.4489
9	consistency	0.3496	0.0036	0.4479
10	sdic	0.1016	0.0011	0.4335

5.3 RQ3 (User Input: Requirement)

To calculate the forecast trajectory of importance (using $tfidf_t$ of user input term), we check for stationarity first. By grouping the $tfidf_t$ by month instead of years, we are able to collect more datapoints, and this series of $tfidf_t$ is used to check for stationarity and to fit the ARIMA model for forecasting.

Figure 3 shows the rolling mean and rolling standard deviation (by 12 periods/months) across the series for user input requirement. We can see that stationarity is confirmed as there is no indication of an upward or downward trend.

Results from the ADF test also confirm this:

- ADF statistic (-5.514) is lower than the critical values at 99% (-3.544), 95% (-2.911) and 90% (-2.593).
- The p-value is also lower than 0.05, so we can reject our null hypothesis and conclude that our data series is stationary.

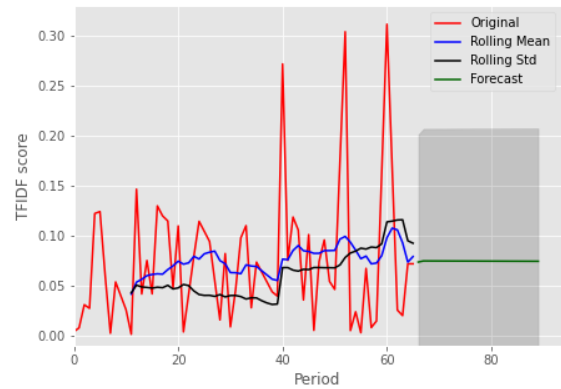


Figure 3: Forecast of $tfidf_t$ values of requirement for the next 24 periods (months).

Fitting the ARIMA model into our series, we can see the trajectory of requirement for the next 24 periods (months). Results in the same Figure 3 show that we would expect a uniform horizontal $tfidf_t$ of range $-0.054 < tfidf_t < 0.207$. As $tfidf_t$ cannot be below zero, we disregard the negative values in the range.

5.4 RQ3 (User Input: Design)

We present Figure 4 for design. This figure shows the rolling mean and rolling standard deviation (by 12 periods/months) across the series. We can see that stationarity is confirmed as there is no indication of an upward or downward trend.

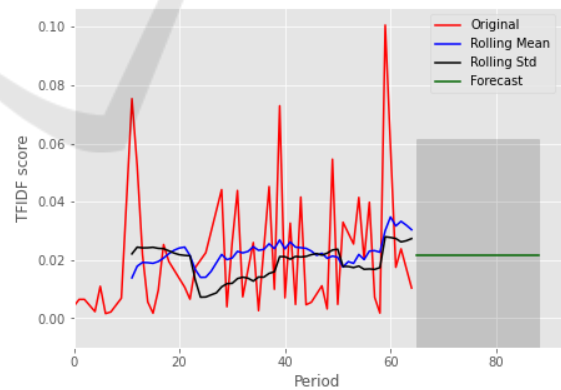


Figure 4: Forecast of $tfidf_t$ values of design for the next 24 periods (months).

Results from the ADF test also confirm this:

- ADF statistic (-6.851) is lower than the critical values at 99% (-3.537), 95% (-2.908) and 90% (-2.591).
- The p-value is also lower than 0.05, so we can reject our null hypothesis and conclude that our data series is stationary.

Results in the same Figure 4 show that we would expect a uniformed horizontal $tfidf_t$ for *design* within range $-0.018 < tfidf_t < 0.0614$. As $tfidf_t$ cannot be below zero, we disregard the negative values in the range.

6 DISCUSSION

At the text processing stage, the content of every paper had to be cleaned to remove noise. The cleaned text was then lemmatised: a process of transforming the morphology of each word to its root form (e.g. *best* → *good*). This allows a more accurate $tfidf_t$ representation of our terms because multiple forms of the same root word should only be considered as one term. Stemming, on the other hand, is a process of truncating suffixes (e.g. *computer* → *comput*). This step was not included in the pipeline for this paper as truncated versions of the terms were not valid in our pre-trained language model vocabulary for semantic filtering. One key limitation of lemmatisers, however, is that not all suffixes are removed. Table 2 shows *architectural* and *architecture* showing the highest similarity to *design*, although they both refer to the same root form *architecture*. In this scenario, stemming would help as it would truncate both terms to *architectur*.

From the $tfidf_t$ of both *requirement* and *design*, we can see the trend of importance throughout recent years, as shown in Figure 2. For *requirement*, we can see a spike in 2018 and 2019. *design*, on the other hand, has a steadily increasing line of importance, in terms of $tfidf_t$. Both of these terms are identified as top-ranked unigrams from our dataset, and more importantly, it shows that both of these SDLC phases play a critical role in traceability using NLP.

Table 1 shows the top 10 semantically similar terms to requirement traceability and Table 2 for design traceability. These semantically related terms allow us to analyse what terms play a part in these main topics (terms) and their importance through its sum $tfidf_t$ and mean $tfidf_t$.

The ranked top 3 for *requirement*: *hipaa*, *regulation* and *constraint*. It appears that during recent years, traceability efforts in requirements relating to HIPAA regulations (Atchinson and Fox, 1997)¹³ play a major role in requirements traceability. Critical software and systems in the healthcare industry necessitate the need for mandatory traceability, and the use of NLP in these traceability efforts is identified as *important*. The top-ranked terms related to

¹³Health Insurance Portability and Accountability Act of 1996

design in Table 2 are about architecture and some related to object-oriented programming concepts, such as *modularization* and *compositional*.

Looking beyond the top 5 terms in Tables 1 and 2, we can observe some terms that are *ambiguous* and may be a part of a compound term (bi-gram or tri-gram) instead of being a part of the user input it belongs to. For example, *case* in requirement traceability may refer to “use case” as a requirement or a “test case” in the SDLC testing phase. There are also overlapping terms between these two SDLC phases, and this confirms that efforts in traceability using NLP typically involve multiple SDLC phases.

In answering RQ3, we present Figures 3 and 4: showing the forecast trajectory of $tfidf_t$ for *requirement* and *design* respectively. Both figures show a horizontal trajectory, which is no surprise, given the seeming stationarity feature in its original form. Despite the spikes we see in some historical datapoints, the $tfidf_t$ for *requirement* is expected to have a maximum boundary of 0.2 for the next 2 years. For *design*, the maximum expected $tfidf_t$ is 0.06 for the next 2 years.

6.1 Implications and Outlook

Our proposed solution has practical and direct implications for the research community, particularly in mining publications to harness value through predictive analytics. We can also target publication venues (conferences etc.). For example, mining published papers from previous conference proceedings (including the past editions of ENASE) to forecast what research areas are expected to emerge next in the field of novel approaches to software engineering. This would particularly be useful to the organising committee and reviewers to better understand the research trajectory in the specified areas of interest, and how these efforts are distributed. This also gives an insight into how a program committee could be formed based on trends and expertise. Since systematic reviews already cover a wide coverage in summarising past papers (albeit with a time lag), this would be particularly useful in ensuring comprehensiveness in content coverage.

For practitioners in the software engineering community, datasets that are rich with free text would equally benefit greatly from our proposed approach. Examples include mining forum posts and Q&A websites (Stack Overflow etc.) to analyse and predict the importance of interest topics. In the context of systematic reviews (i.e., what was presented in this paper), the connection between systematic reviews and software engineering practice is arguably lacking

and insufficient (Cartaxo et al., 2016; da Silva et al., 2011). This was also investigated in a recent paper by analysing how well systematic reviews help to answer questions posted online through data from Q&A websites (Cartaxo et al., 2017). With the vast amount of free text being made available to the masses (coupled with timestamps), our position paper proposes a combination of NLP and time series that can benefit practitioners by improving the connection (through analysis of historical and future predictions) between systematic reviews and real-world problems in the software engineering community.

6.2 Threats to Validity

- *External validity:* We acknowledge the limited number and narrow topic scope of datapoints. However, we present this work to showcase the applicability of industry practices (as the case studies presented in Section 1) in forecasting emerging trends albeit within the scope of systematically reviewing the literature.
- *Internal validity:* Forecasting any prediction requires extensive analysis of multiple factors that may influence expected results. A key limitation of ARIMA is that it relies on the assumption that past values of the series can alone predict future values. Regardless, being aware of this does not negate the importance of ARIMA in forecasting predictions.
- *Conclusion validity:* Extracted raw data (PDF text) consisting of non-uniform structure, resulting in less accurate observations and conclusions. There is also some degree of non-uniformity in the way the published text represents meaning. For example, the use of synonym terms that relate to one concept will dilute the $tfidf_i$ of that concept we are interested in (by user input).

7 CONCLUSION AND FUTURE WORK

In this position paper, we introduced a meta-analysis of publications in the form of predictive analysis, through forecasting emerging research areas from systematic studies. We combined NLP techniques with time series analysis to detect and identify the trend of a subset research area, denoted by its term.

Our approach leverages recent advancements in language models and time series. It expands the analysis of systematic reviews in a particular research focus, and its applicability can be used for other re-

search areas as well. Based on the papers from our own SMS, we showed that our proposed approach can (i) detect trends of importance within the terms emerging from the literature, and (ii) forecast trends in research based on the results of a specific systematic study. This paper is a glimpse into the prospects of leveraging data science for software engineering academic purposes.

An extension of our methodology would be to go further in analysing semantically related terms. We can observe that not all terms derived are of similar hierarchy (in terms of the topic scope), and constructing a taxonomy based on existing defined ontology (such as the ACM Computing Classification System (Cassel et al., 2013)) will allow us to match and understand directions and categories of research efforts more effectively.

With regards to the time series analysis, future works involve testing out model results to validate predictions: using a data series from the previous decade to validate the results against the current one. This helps to see how the solution was able to identify and detect the trend of importance for terms that may have been gaining traction in the training set. For example, in deep learning NLP, BERT was only introduced and published in 2018 (Devlin et al., 2018). If we were to look into previous papers on the topic of deep learning and transformers, we should expect relevant important terms to emerge such as neural network and transfer learning.

In the software engineering community, emerging solutions from the field of data science have taken the limelight in recent times for researchers and practitioners. Our paper explores this by analysing *leading indicators* rather than lagging ones – those which systematic reviews typically focus on. The sheer advancement of language models and data analytics in recent years have made giant leaps of progress and capitalising on these would be the *holy grail*.

REFERENCES

- Ananiadou, S., Rea, B., Okazaki, N., Procter, R., and Thomas, J. (2009). Supporting systematic reviews using text mining. *Social Science Computer Review*, 27(4):509–523.
- Asooja, K., Bordea, G., Vulcu, G., and Buitelaar, P. (2016). Forecasting emerging trends from scientific literature. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 417–420, Portorož, Slovenia. European Language Resources Association (ELRA).
- Atchinson, B. K. and Fox, D. M. (1997). From the field:

- The politics of the health insurance portability and accountability act. *Health Affairs*, 16(3):146–150.
- Beel, J., Gipp, B., Langer, S., and Breiting, C. (2016). Research-paper recommender systems : a literature survey. *International Journal on Digital Libraries*, 17(4):305–338.
- Bilgili, F. (1998). Stationarity and cointegration tests: Comparison of engle-granger and johansen methodologies. *Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, (13):131–141.
- Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583.
- Cartaxo, B., Pinto, G., Ribeiro, D., Kamei, F., Santos, R. E., da Silva, F. Q., and Soares, S. (2017). Using q&a websites as a method for assessing systematic reviews. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 238–242.
- Cartaxo, B., Pinto, G., Vieira, E., and Soares, S. (2016). Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16*, New York, NY, USA. Association for Computing Machinery.
- Cassel, L. N., Palivela, S., Marepalli, S., Padyala, A., Deep, R., and Terala, S. (2013). The new acm ces and a computing ontology. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13*, page 427–428, New York, NY, USA. Association for Computing Machinery.
- da Silva, F. Q., Santos, A. L., Soares, S., França, A. C. C., Monteiro, C. V., and Maciel, F. F. (2011). Six years of systematic literature reviews in software engineering: An updated tertiary study. *Information and Software Technology*, 53(9):899–913. Studying work practices in Global Software Engineering.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.
- Efstathiou, V., Chatzilenas, C., and Spinellis, D. (2018). Word embeddings for the software engineering domain. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 38–41.
- Kitchenham, B., Dyba, T., and Jorgensen, M. (2004). Evidence-based software engineering. In *Proceedings. 26th International Conference on Software Engineering*, pages 273–281.
- Li, X., Xie, Q., Jiang, J., Zhou, Y., and Huang, L. (2019). Identifying and monitoring the development trends of emerging technologies using patent analysis and twitter data mining: The case of perovskite solar cell technology. *Technological Forecasting and Social Change*, 146:687–705.
- Marshall, C. and Brereton, P. (2015). Systematic review toolbox: A catalogue of tools to support systematic reviews. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE '15*, New York, NY, USA. Association for Computing Machinery.
- Marshall, C., Brereton, P., and Kitchenham, B. (2015). Tools to support systematic reviews in software engineering: A cross-domain survey using semi-structured interviews. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE '15*, New York, NY, USA. Association for Computing Machinery.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
- Olorisade, B. K., de Quincey, E., Brereton, P., and Andras, P. (2016). A critical analysis of studies that address the use of text mining for citation screening in systematic reviews. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16*, New York, NY, USA. Association for Computing Machinery.
- O'Mara-Eves, A., Thomas, J., McNaught, J., Miwa, M., and Ananiadou, S. (2015). Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Syst. Rev.*, 4(1):5.
- Osborne, F., Muccini, H., Lago, P., and Motta, E. (2019). Reducing the effort for systematic reviews in software engineering. *Data Science*, 2(1-2):311–340.
- Pauzi, Z. and Capiluppi, A. (2023). Applications of natural language processing in software traceability: A systematic mapping study. *Journal of Systems and Software*, 198. Publisher Copyright: © 2023 The Author(s).
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Stansfield, C., O'Mara-Eves, A., and Thomas, J. (2017). Text mining for search term development in systematic reviewing: A discussion of some methods and challenges. *Research Synthesis Methods*, 8(3):355–365.
- Thomas, J., McNaught, J., and Ananiadou, S. (2011). Applications of text mining within systematic reviews. *Research Synthesis Methods*, 2(1):1–14.
- Zhang, H. and Ali Babar, M. (2013). Systematic reviews in software engineering: An empirical investigation. *Information and Software Technology*, 55(7):1341–1354.