# DRIFT: A Toolkit for Diachronic Analysis of Scientific Literature

**Abheesht Sharma***
Dept. of CS&IS
BITS Pilani, Goa Campus
f20171014@goa.bits-pilani.ac.in

**Gunjan Chhablani***
Dept. of CS&IS
BITS Pilani, Goa Campus
chhablani.gunjan@gmail.com

**Harshit Pandey***
Dept. of Computer Science
Pune University
hp2pandey1@gmail.com

**Rajaswa Patil**
Dept. of E & E Engineering
BITS Pilani, Goa Campus
f20170334@goa.bits-pilani.ac.in

## Abstract

In this work, we present to the NLP community, and to the wider research community as a whole, an application for the diachronic analysis of research corpora. We open source an easy-to-use tool coined *DRIFT*, which allows researchers to track research trends and development over the years. The analysis methods are collated from well-cited research works, with a few of our own methods added for good measure. Succinctly put, some of the analysis methods are: keyword extraction, word clouds, predicting declining/stagnant/growing trends using Productivity, tracking bi-grams using Acceleration plots, finding the Semantic Drift of words, tracking trends using similarity, etc. To demonstrate the utility and efficacy of our tool, we perform a case study on the *cs.CL* corpus of the arXiv repository and draw inferences from the analysis methods. The toolkit and the associated code are available here.

## 1 Introduction

Historians perform comparative studies between the past and the present to explain certain phenomena. Studying the past also helps us in making plausible predictions about the future. Major sources of information about the past are old-age texts, tomes and manuscripts. Language changes and shifts with changes in society and culture. For example, words such as *curglaff* and *lunting* have become obsolete, the word *gay*'s meaning has changed entirely and new words such as *LOL* and *ROFL* have gained prominence with the emergence of the "texting generation".

In the research world, analysis of old articles and papers is gaining importance. Analysing old documents can revive research topics which have been forgotten over the decades; removing the cobwebs on these topics can inspire path-breaking research. Explanations of why we arrived at certain conclusions can also be gleaned by peeping into the past. Performing such diachronic analysis of text has become easy because of the rapid growth of NLP. Temporal word embeddings such as TWEC (Di Carlo et al., 2019) enable us to do so.

In this paper, we introduce a hassle-free application for the diachronic analysis of research corpora. We name our application *DRIFT*, an acronym for **D**iach**R**onic Analysis of Scient**IF**ic Li**T**erature. *DRIFT* provides researchers a one-click way to perform various diachronic analyses.

Our contribution is two-fold: 1) We assemble (and propose) different methods for diachronic analysis of research corpora; 2) We make it extremely convenient for researchers to analyse research trends; this, in turn, will encourage researchers to find latent, quiescent topics which may have huge research potential in the near future. The main principles behind the design for *DRIFT* are **ease of usage** and **flexibility**.

The rest of the paper is organised as follows. In Section 2, we perform an extensive literature survey. Section 3 describes the research corpora and the methods for crawling the corpora. In Section 4, we explain the training methodology for obtaining diachronic embeddings and our easy-to-use application's dashboard and layout in detail. Section 5 elucidates the various analysis methods which researchers can employ with a mere click of a button.

## 2 Related Works

### 2.1 Diachronic Embeddings

Savants have attempted to leverage word embeddings to study semantic changes of words over time (Hamilton et al., 2016; Kulkarni et al., 2014).

* Equal contribution. Author ordering determined by coin flip.



*Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 361–371

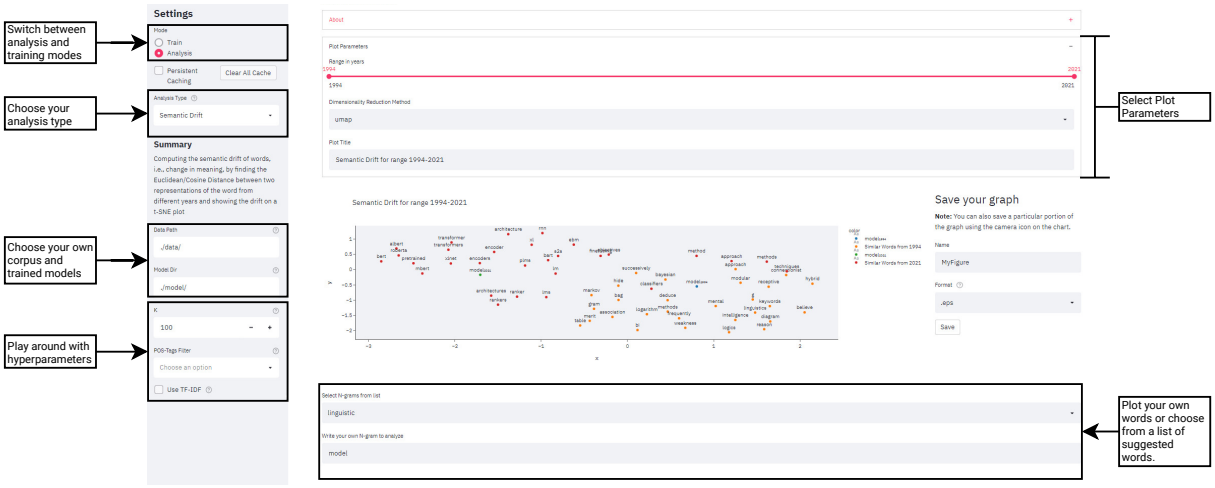



Figure 1: Snapshot for the application dashboard.

The major issue has been the inability to compare word vectors of two different time periods since word embedding algorithms are stochastic in nature and invariant under rotation, i.e., they belong to different coordinate systems (Kutuzov et al., 2018). Therefore, attempts have been made to align these vectors (Szymanski, 2017; Schlechtweg et al., 2019; Bamman et al., 2014). The most recent attempt to resolve this issue is TWEC (Di Carlo et al., 2019), which implicitly aligns the word vectors. We use TWEC in our work.

### 2.2 Analysis Attempts and Tools for Research Corpora

Many researchers have devoted their time to analysing different research corpora with respect to time. Most of the analysis is citation-based, or the researchers analyse the diversity of the research world (age-based, gender-based analysis) (Mohammad, 2019). In some, authors have introduced new research corpora such as the NLP4NLP Corpus (Mariani et al., 2019). Gollapalli and Li (2015) perform comparative analyses between two conferences such as EMNLP and ACL by expressing both venues as a probability distribution over time. A good amount of research has also focused on statistical techniques. For example, Schumann (2016) uses frequency and productivity to identify emerging/stagnant/falling topics. Others have devised Machine Learning/Deep Learning methods to identify future trends (Francopoulo et al., 2016).

NLP Scholar (Mohammad, 2020) is an interactive, visual tool which makes it simpler for researchers to find impactful scientific articles in bygone years from the ACL Anthology based on the number of citations, searching for relevant related works, study the changes in the number of articles and citations, etc.

While significant effort has gone into procuring research corpora and in devising different analysis methods, not much work has been done in building a user-friendly, convenient application which ties up statistical, learning-based analysis and temporal embedding-based methods and makes it easy for the research community to draw inferences and identify research trends.

## 3 Datasets

arXiv[1] is a free-access repository of e-prints of scientific papers, articles, essays and studies, launched in 1991 by Cornell University. arXiv covers a vast array of disciplines such as Computer Science, Mathematics, Physics, Chemistry, etc. We perform our experiments and analysis on a burgeoning subdomain of Computer Science, namely, Computation and Language (*cs.CL*).

Moreover, we perform our analyses on abstracts. The rationale behind this is: 1) The abstract of a paper condenses and encapsulates the whole paper and conveys its major contributions; 2) The rest of the paper may have tables, figures, arcane mathematical equations, intricacies and esoteric jargon which are difficult to work with.

We restrict our analysis to the 1994-2021 period for the following reasons: 1) arXiv has cs.CL and cs.CV papers from 1994 onwards; 2) We considered crawling the abstracts of papers published prior to 1994 from their PDFs (we obtained the

[1] https://arxiv.org/

URLs from the ACL Anthology[2]). However, this proved to be no mean feat, considering that the old PDFs used different font styles, unselectable texts, etc. The procured abstracts had grammatical flaws.

We use the arXiv API to obtain the metadata of the papers (specifically, URL, date of submission, title, authors, and abstract). To ensure that low quality articles are not included, we sort the papers by their relevance. In total, we analyse 27,384 abstracts.

## 4 Methodology

### 4.1 Diachronic Embeddings

As mentioned earlier, we use TWEC (Di Carlo et al., 2019) (Temporal Word Embeddings with Compass) to create dynamic word embeddings for each time slice. TWEC uses Word2Vec (Mikolov et al., 2013) and trains it on data for all the years to learn atemporal embeddings. Then, the "pre-trained" target embeddings are frozen, which serves as the "compass" and temporal embeddings are learned for each year as context embeddings. We use this method because it scales well with large corpora, and it also makes the training simple and efficient.

### 4.2 Application

We build our application using Streamlit[3], a framework in Python for making data applications. An overview of the application dashboard is shown in Figure 1. The dashboard has two modes: *Train* and *Analysis*. The general workflow of the application is shown in Figure 2 and described as follows:

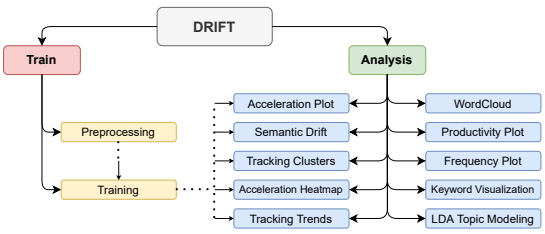



Figure 2: System design for the application.

#### 4.2.1 *Train* Mode

For flexibility, in the *Train* mode, we give users an option to upload their corpus and train the TWEC model with a click of a button. The sidebar has two subsections.

**Preprocessing** Before training TWEC, the dataset has to be preprocessed. The preprocessing pipeline we employ comprises simple steps: convert to lower-case, lemmatisation, remove punctuation and stopwords, remove non-alphanumerics, etc. We also remove words which frequently appear in research papers such as *paper*, *systems*, *result*, *approach*, etc. This subsection has three input text boxes: *JSON Path* for the input dataset/corpus, *Text Key* for the key whose corresponding value contains raw text, *Data Path*, where the preprocessed data is to be stored. On clicking the button *Preprocess*, the raw text is preprocessed.

**Training** TWEC hyperparameters such as embedding size, number of static iterations, dynamic iterations, negative samples, window size are presented as text boxes/drop-down menus here. On clicking the button *Train*, the TWEC model is trained on the corpus. Options/hyperparameters/input paths for preprocessing and TWEC training are presented as text boxes/drop-down menus in the sidebar.

#### 4.2.2 *Analysis* Mode

The *Analysis* mode is the most vital component of the application. The analysis method can be chosen from a drop-down menu. Once an analysis method is chosen, the sidebar displays several parameters which generally include data path, K (number of keywords to choose from compass), whether to use TF-IDF or use particular Parts-of-Speech, etc. As an example, method-specific parameters for *Semantic Drift* include: model path, K(sim.) (top-k most similar words to be shown around chosen words), K(drift) (allows user to select from top-k most drifted words), and distance metric (euclidean/cosine distance).

The page for every analysis method contains an expander, i.e., a collapsible section which has options such as a slider to decide the range of years to be considered for analysis, or a text box for a particular word to be analysed. Below the expander, interactive graphs, t-SNE (Maaten and Hinton, 2008)/UMAP (McInnes et al., 2020)/PCA plots and tables are displayed, based upon user's selection. Every button, text box, drop-down menu has a "help" option to guide users on its usage and purpose.

On the right-hand side, there is an option to export graphs in different formats such as PDF, SVG, etc. for easy integration with research papers.

[2] https://github.com/acl-org/acl-anthology
[3] https://streamlit.io/

## 5 Analysis Methods and Discussion

### 5.1 Word Cloud

A Word Cloud is a graphical representation of the keywords of a corpus, i.e., words which have higher frequency are given more importance. This importance is translated in terms of size and colour in the visualisation. We give the user an option to choose the year for which the analysis is to be done. Other options in the sidebar include minimum and maximum font size, number of words to be displayed, colour, width and height of the word cloud. The user can select the year from the expander. On the main page, we display the word cloud. An example is shown in Figure 7.

### 5.2 Productivity/Frequency Plot

Schumann (2016) uses normalized term frequency and term productivity as measures for identifying growing/consolidated/declining terms. Term productivity is a measure of the ability of a concept to produce new multi-word terms. In our case, we use bigrams. For each year $y$ and single-word term $t$, and associated $n$ multi-word terms $m$, the productivity is given by the entropy:

$$e(t, y) = -\sum_{i=1}^{n} \log_2(p_{m_i,y}).p_{m_i,y} \quad (1)$$

where

$$p_{m,y} = \frac{f(m)}{\sum_{i=1}^{n} f(m_i)} \quad (2)$$

and $f(m)$ is frequency of the term. Based on these two measures (over the years), words are clustered into three categories:

- **Growing Terms**: Those which have increasing frequency and productivity in the recent years.
- **Consolidated Terms**: Those that are growing in frequency, but not in productivity.
- **Terms in Decline**: Those which have reached an upper bound of productivity and have low frequency.

In the sidebar, the user can choose the $N$ in N-gram and the $K$ in Top-K. The default for keyword extraction in all the methods is normalised frequency, but the user can opt for frequency or TF-IDF. Alternatively, the user can also choose the POS tag for filtering the keywords.

In the expander, the user can select words from the suggested keywords or add their own words for analysis. In the main section, the productivity and normalised frequency graphs are displayed. Below these graphs, we have a dataframe which displays which clusters the words belong to. An example is shown in Figure 10.

### 5.3 Acceleration Plot

Inspired by Dridi et al. (2019), we analyse the semantic shift of a pair of words with respect to each other. We aim to identify fast converging keywords using "acceleration". The acceleration matrix is calculated as the difference between similarity matrices of two different time periods, where each entry is:

$$acc(w_i, w_j)^{t\to(t+1)} = sim(w_i, w_j)^{(t+1)} - sim(w_i, w_j)^{t} \quad (3)$$

where $(w_i, w_j)$ is the word pair being analysed, $sim(w_i, w_j)$ is the cosine similarity between words $w_i$ and $w_j$, and $(t, t+1)$ is the time range. If the words $w_i$ and $w_j$ converge, the cosine similarity value between them will increase, or in other words, $acc(w_i, w_j)^{t\to(t+1)} > 0$. We pick the word pairs with the highest acceleration values.

To demonstrate the convergence of a pair of words, we plot the pair on a two-dimensional plot, along with the top-K most similar words around them. An example is shown in Figure 4.

In addition to $K$, POS Tag Filters and an option to opt for TF-IDF, we also have $K(acc.)$ and $K(sim.)$. The parameter, $K(sim.)$, is for deciding how many words to plot around the two chosen words. $K(acc.)$ is for finding the top-K most accelerated pair of words.

The expander has a slider for selecting the range of years, below which there is a dataframe, listing the top-K most accelerated keywords. There are two drop-downs for selecting the pair of words to be analysed.

The main section has the graph of the embedding space, showing the convergence/divergence of the chosen pair of words.

### 5.4 Semantic Drift

The semantics of words perpetually change over time, i.e., words drift from one point to another. This incessant *semantic drift* is a product of changing social and cultural norms, transition in linguistic usage, amongst other factors. Since we use

| | model$_{2010}$ | gmms$_{2011}$ | markov$_{2014}$ | hmm$_{2016}$ | asr$_{2017}$ | acoustic$_{2019}$ |
|---|---|---|---|---|---|---|
| 0 | gmms$_{2011}$ (0.6) | subspace$_{2014}$ (0.78) | hmm$_{2016}$ (0.68) | markov$_{2017}$ (0.61) | e2e$_{2020}$ (0.5) | phonetic$_{2021}$ (0.5) |
| 1 | gaussian$_{2011}$ (0.51) | mixture$_{2014}$ (0.71) | hmms$_{2016}$ (0.66) | dnn$_{2017}$ (0.59) | hmm$_{2020}$ (0.49) | spectrograms$_{2020}$ (0.47) |
| 2 | hide$_{2011}$ (0.51) | gaussian$_{2014}$ (0.69) | dnn$_{2016}$ (0.55) | gmm$_{2017}$ (0.58) | acoustic$_{2019}$ (0.48) | audio$_{2021}$ (0.46) |
| 3 | mixture$_{2011}$ (0.5) | markov$_{2014}$ (0.67) | conditional$_{2016}$ (0.55) | hmms$_{2017}$ (0.56) | transcription$_{2018}$ (0.47) | pgs$_{2020}$ (0.46) |
| 4 | allocation$_{2013}$ (0.47) | factorial$_{2013}$ (0.66) | cd$_{2015}$ (0.53) | asr$_{2017}$ (0.52) | transcriptions$_{2019}$ (0.47) | mfcc$_{2020}$ (0.45) |

Figure 3: Tracking the word *"model"* from the year 2010 to the year 2019 with stride=3.

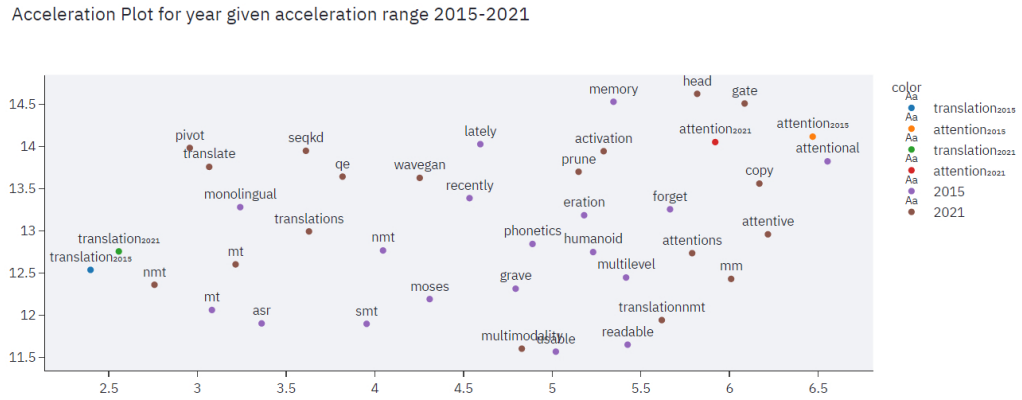



Figure 4: Convergence of the words: *"Translation"* and *"Attention"* (2015-2021).

TWEC embeddings, the representations of a word across time periods are implicitly aligned and we can directly compute the shift as the distance between the two representations.

We compute the drift as the Euclidean Distance or the Cosine Distance between the first year embedding of the word and the last year embedding of the word. We sort the words according to their drift (in descending order). For more details on the algorithm, refer to Subsection A.1 in the Appendix.

In order to visualise the drift, we plot the two representations of the word on a UMAP/tSNE/PCA plot, along with the most similar words in the respective years which are clumped around the two representations. Plotting the top-k most similar words helps the user in analysing the shift in meaning.

Other than the usual parameters - $K$, POS Tag Filters, TF-IDF, the sidebar has options to choose $K(sim.)$ and $K(drift)$. The expander has a slider for selecting the range of years. Below the graphs, a drop-down menu has the top-K(drift) most drifted words as suggestions. The user can also input a custom word.

A couple of examples are shown in Figure 5. Clearly, the word "model" has drifted from the vicinity of *bayesian*, *markov*, *bag*, *gram*, *logics* in 2017 to *albert*, *roberta*, *xlnet*, *mbert*, *bart* in 2021. This makes sense, because back in 1994, a conventional NLP model was related to bag-of-words, and Markov/HMM-based (Fosler-Lussier, 1998)/Bayesian methods, while in 2021, the focus has shifted to RoBERTa (Liu et al., 2020), XL-Net (Yang et al., 2019), BART (Lewis et al., 2020) and other transformer models.

### 5.5 Tracking Clusters

Word meanings change over time. They come closer or drift apart. In any given year, certain words are clumped together, i.e., they belong to one cluster. But over time, clusters can break into two/coalesce together to form one. Unlike the previous module which tracks movement of one word at a time, here, we track the movement of clusters.

Our implementation is simple. Given a range of years, and some common keywords, we cluster the words for every year, and display the UMAP/tSNE/PCA graph (with clusters) for every year separately. We use the KMeans Clustering Algorithm. A visualisation is shown in Figure 9.

The additional options in the sidebar include number of clusters and max. number of clusters. If number of clusters is set to 0, the application finds the optimal number by searching between $[3, Max.\ Number\ of\ Clusters]$ using Silhouette score. Additionally, we have an option to choose the library of implementation: faiss[4]/scikit-learn[5]. Faiss is up to 10 times faster than sklearn.

[4] https://github.com/facebookresearch/faiss

[5] https://github.com/scikit-learn/scikit-learn

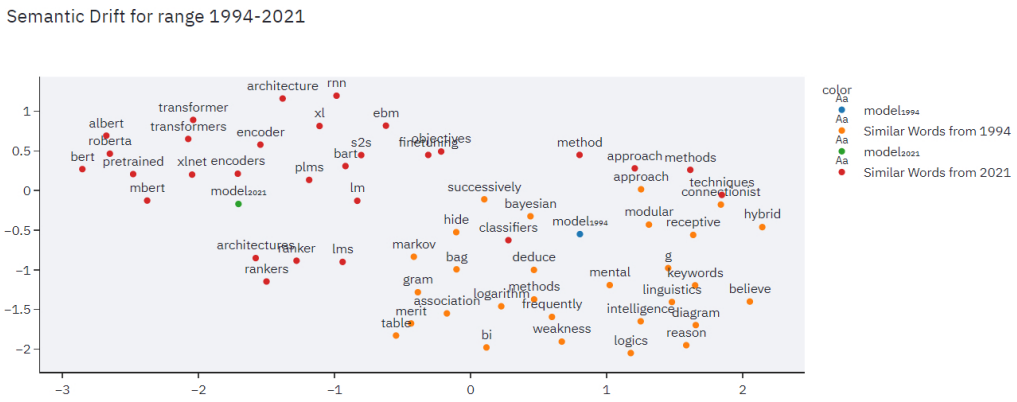



Figure 5: Semantic Drift of the word *"model"* (1994-2021).

### 5.6 Acceleration Heatmap

We use the same *acceleration* measure as in Section 5.3. Instead of listing the top-K pairs based on acceleration, we plot a heatmap of the acceleration between two years. This can be useful in analysing how different word-pairs are converging at different rates between two years. This heatmap can be used in combination with the Acceleration Plot, where a representation of the word-pairs selected based on the heatmap can be explored. Refer to Figure 6 to get an idea of the visualisation.

### 5.7 Track Trends with Similarity

We wish to chart the trajectory of a word/topic from *year 1* to *year 2*. To accomplish this, we allow the user to pick a word from *year 1*. At the same time, we ask the user to provide the desired *stride*. We search for the most similar word in the next $stride$ years. We keep doing this iteratively till we reach *year 2*, updating the word at each step via user input from the top-K similar words. An overview of the algorithm is given in Subsection A.2. An example is given in Figure 3. The trajectory we follow is model, gmms, markov, hmm, asr, phonetic. This makes sense, because HMMs, GMMs were used for ASR in the past.

### 5.8 Keyword Visualisation

Here, we use the YAKE Keyword Extraction method (Campos et al., 2020) to extract keywords. Yake Score is indirectly proportional to the keyword importance. Hence, we report the scores differently[6].

The user can select $Max.\ N-gram$ in the sidebar. The year can be selected from the slider in the expander. The main section has a bar graph, with n-grams on the y-axis and their importance scores on the x-axis. A visualisation is shown in Figure 11.

### 5.9 LDA Topic Modelling

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a generative probabilistic model for an assortment of documents, generally used for topic modelling and extraction. LDA clusters the text data into imaginary topics. Every topic can be represented as a probability distribution over n-grams and every document can be represented as a probability distribution over these generated topics.

We train LDA on a corpus where each document contains the abstracts of a particular year. We express every year as a probability distribution of topics. An example is shown in Figure 8.

## 6 Accessibility

The code for the toolkit is open-sourced, and is distributed under a MIT License. The latest stable code release is available here. An online demo of the application is hosted here. A set of short video demonstrations can be found here.

## 7 Conclusion and Future Work

In this paper, we present *DRIFT*, a hassle-free, user-friendly application for diachronic analysis of scientific literature. We compile well-cited research works and provide an interactive and intuitive user-interface using Streamlit. We perform a case-study on *cs.CL* corpus from arXiv repository, and demonstrate the effectiveness and utility of our application in analysing such corpora. As an extension of our work, apart from adding upcoming analysis methods (citation-based/statistical/knowledge graphs), we also intend to make our application

[6] $new\_score = \frac{1}{10^5 \times yake\_score}$

modular to make it easier for users to add their own methods, provide semi-automated inference from the graphs/tables, and allow easy integration with LaTeX.

## References


David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834, Baltimore, Maryland. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Valerio Di Carlo, Federico Bianchi, and Matteo Palmonari. 2019. Training temporal word embeddings with a compass. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6326–6334.

Amna Dridi, Mohamed Medhat Gaber, R. Muhammad Atif Azad, and Jagdev Bhogal. 2019. Deephist: Towards a deep learning-based computational history of trends in the nips. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Eric Fosler-Lussier. 1998. Markov models and hidden markov models: A brief tutorial. *International Computer Science Institute*.

Gil Francopoulo, Joseph Mariani, and Patrick Paroubek. 2016. Predictive modeling: Guessing the NLP terms of tomorrow. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 336–343, Portorož, Slovenia. European Language Resources Association (ELRA).

Sujatha Das Gollapalli and Xiaoli Li. 2015. EMNLP versus ACL: Analyzing NLP research over time. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2002–2006, Lisbon, Portugal. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2014. Statistically significant detection of linguistic change.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

L. V. D. Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Joseph Mariani, Gil Francopoulo, and Patrick Paroubek. 2019. The nlp4nlp corpus (i): 50 years of publication, collaboration and citation in speech and language processing. *Frontiers in Research Metrics and Analytics*, 3:36.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif M. Mohammad. 2019. The state of nlp literature: A diachronic analysis of the acl anthology.

Saif M. Mohammad. 2020. NLP scholar: An interactive visual explorer for natural language processing literature. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 232–255, Online. Association for Computational Linguistics.

Dominik Schlechtweg, Anna Hätty, Marco Del Tredici, and Sabine Schulte im Walde. 2019. A wind of change: Detecting and evaluating lexical semantic change across times and domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 732–746, Florence, Italy. Association for Computational Linguistics.

Anne-Kathrin Schumann. 2016. Brave new world: Uncovering topical dynamics in the ACL Anthology reference corpus using term life cycle information. In *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 448–453, Vancouver, Canada. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

# A Algorithms

## A.1 Semantic Shift

An overview of the approach is given in Algorithm 1. The $dist(x, y)$ function can either be the Euclidean Distance or Cosine Distance. In the algorithm, we find the most drifted words from a given list of words by sorting them according to the distance. $year\_x\_model$ is the aligned TWEC Model for year $x$.

**Algorithm 1:** Semantic Shift

**Data:** $words, year\_1, year\_2$
**Result:** $word\_dist$

$word\_dist \leftarrow \{\}$
**for** $word$ *in* $words$ **do**
  $year\_1\_emb \leftarrow year\_1\_model(word)$
  $year\_2\_emb \leftarrow year\_2\_model(word)$
  $word\_dist[word] \leftarrow dist(year\_1\_emb, year\_2\_emb)$
**end**
$word\_dist \leftarrow sort\_by\_value(word\_dist, \text{“}desc\text{”})$

## A.2 Track Trends with Similarity

Algorithm 2 gives an overview of our approach. $MSW(word, range = [year+1, year+stride])$ finds the most similar word to $word$ in the years $year + 1, year + 2, ..., year + stride$. Here, we have demonstrated (for the sake of simplicity) that the most similar word is chosen at every step. In the actual implementation, we allow the user to choose from the top-K most similar words at every step. $stride$ is available as an argument in the sidebar, along with $K(sim.)$. In the main section, a table (which is dynamically updated) displays the trajectory taken by the initial word.

**Algorithm 2:** Track Trends with Similarity

**Data:** $word, year\_1, year\_2$
**Result:** $word\_traj$

$year \leftarrow year\_1$
$word\_iter \leftarrow word$
$word\_traj \leftarrow [(word, year\_1)]$
**while** $year \leq year\_2$ **do**
  $word\_iter \leftarrow MSW(word\_iter, range = [year + 1, year + stride])$
  $year \leftarrow get\_year(word\_iter)$
  $word\_traj.append((word\_iter, year))$
**end**

# B Additional Demonstrative Analyses

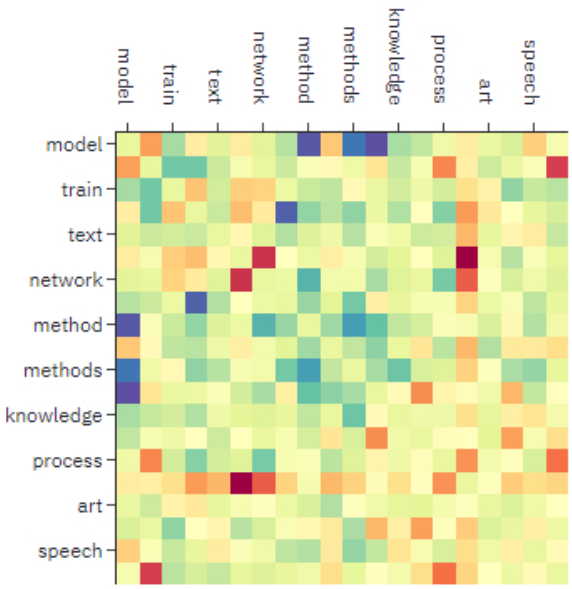



Figure 6: Acceleration Heatmap for top-K words between two years. A darker red colour implies a higher positive acceleration value.


(a) Word Cloud for the year 2000.

(b) Word Cloud for the year 2018.

Figure 7: Word Cloud Demonstrations.

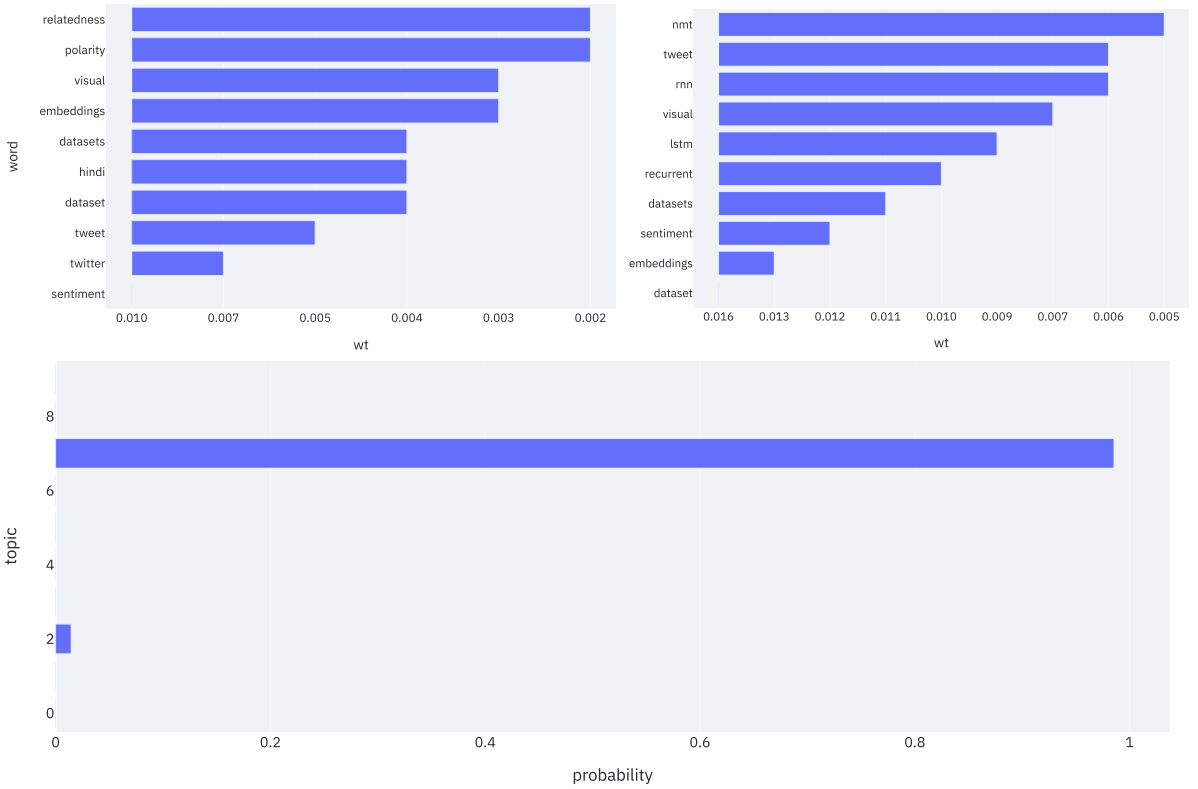
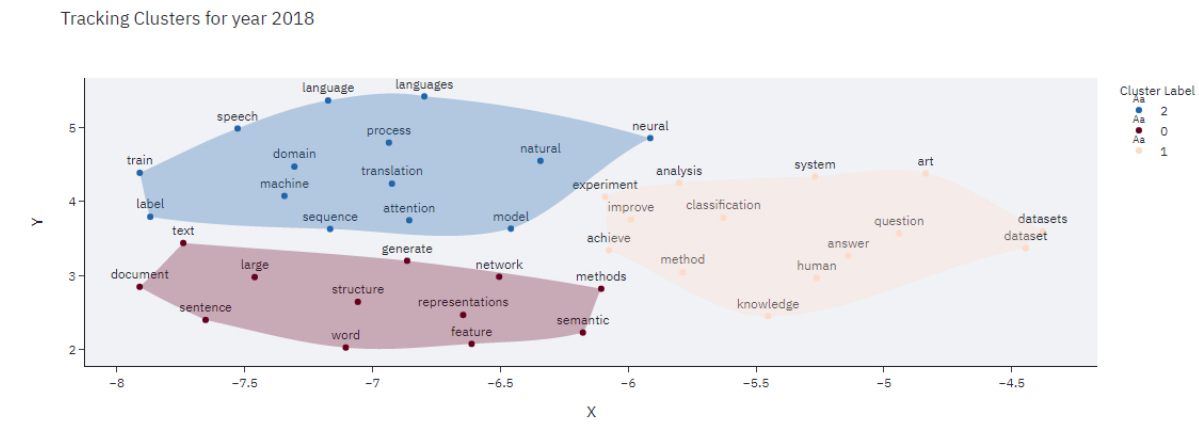
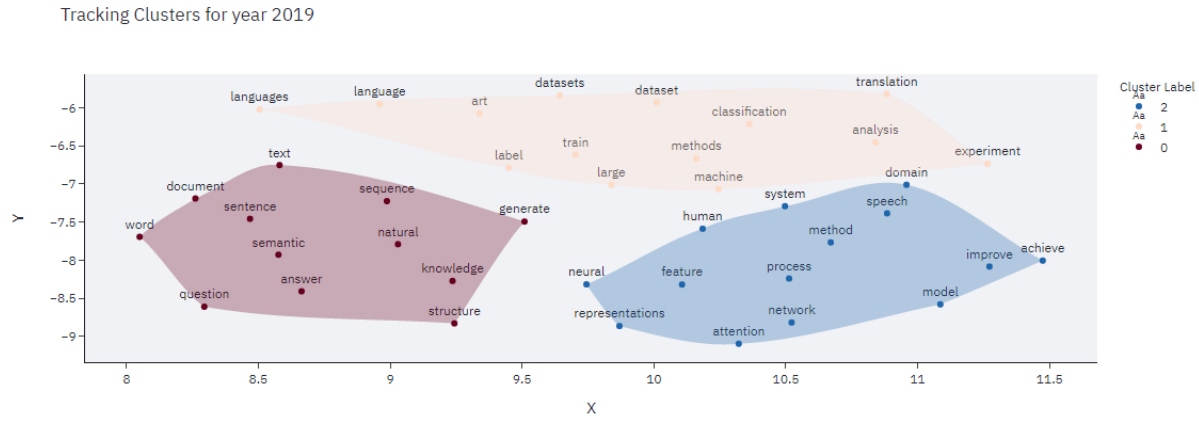



Figure 8: LDA Topic Modeling for the year 2014. The major topics in 2014 are 2 and 7. Topic 2 is primarily composed of “visual”, “embeddings”, etc. and Topic 7 is composed of “nmt”, “tweet”, “rnn”, etc.

Optimal Number of Clusters: 3




Optimal Number of Clusters: 3




Optimal Number of Clusters: 2

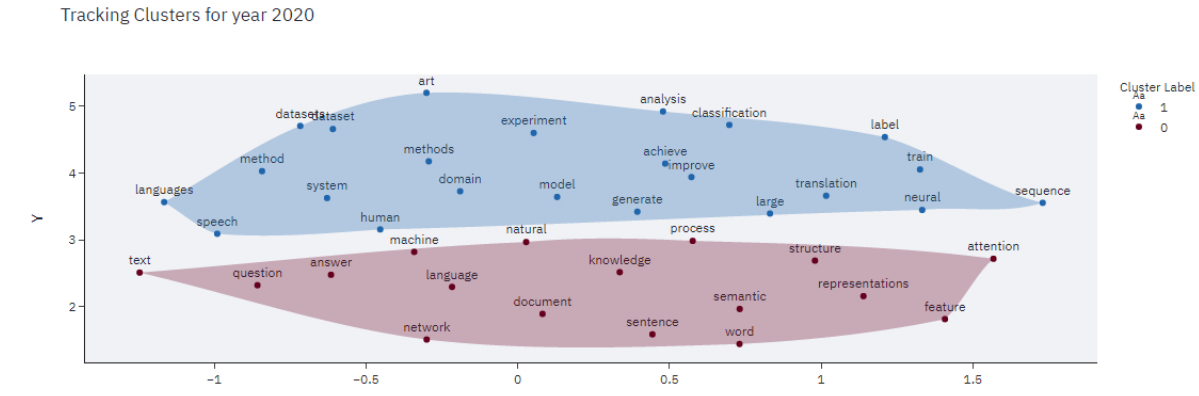



Figure 9: Tracking Clusters from 2018 to 2020.

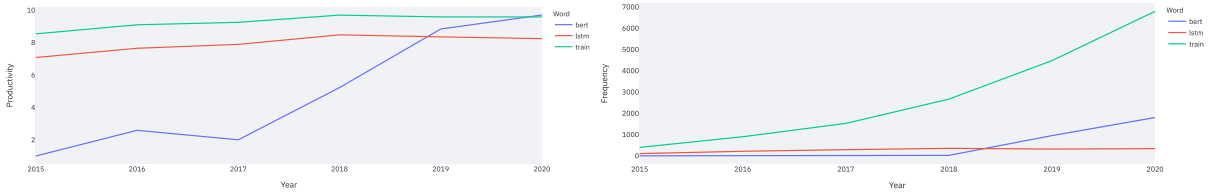



Figure 10: Productivity and Normalised Frequency Plots for the words “BERT”, “LSTM”, “train” (2015-2020). We can classify “BERT” as a growing term, “train” as a consolidated term and “LSTM”, a declining term.

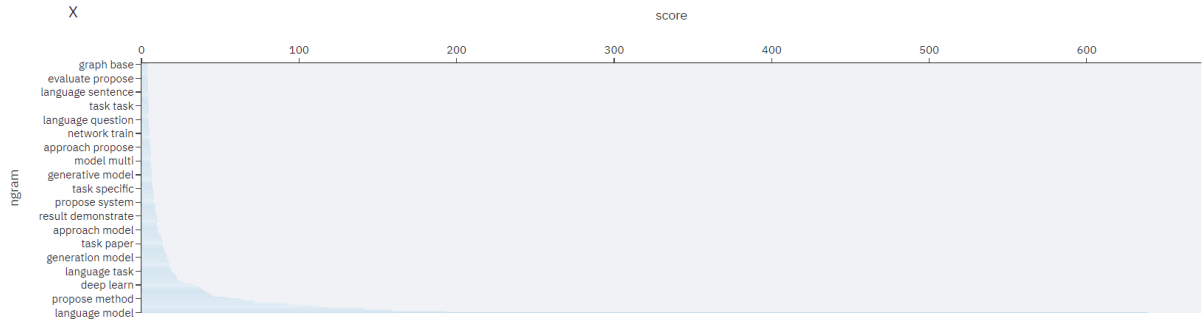



Figure 11: Keyword Visualisation using the YAKE keyword extraction method for the year 2019. Top keywords with $n = 2$ include “language model”, “deep learn”, “generative model”, etc.