A Quantitative Weak Signal Framework for Detecting Emerging Computer Vision Technologies Using arXiv Data

Aditya Agarwal[1], Parneeta Chaudhary[1#], Siddhant Kumar Sharma[1]

School of Engineering & Technology, Vivekananda Institute of Professional Studies- Technical Campus, AU-Block (Outer Ring Road), Pitampura, Delhi - 110034, India.

Parneeta Chaudhary[#]

E-mail: parnitachaudhary@gmail.com ; parneeta@vips.edu

ORCID ID: 0000-0003-4314-3573

Aditya Agarwal

E-mail: aditya.smcs@gmail.com ; 08117702722_cse@vips.edu

ORCID ID: 0009-0005-4200-5795

Siddhant Kumar Sharma

E-mail: siddhantsharma4520@gmail.com ; 02417711922_ds@vips.edu

Online Resource 1: Implementation Details and Software Infrastructure

We believe that scientific research should be transparent and easy for others to build upon. To support this, we have detailed the exact technical setup used in our Weak Signal Analysis framework below. The entire pipeline was built using **Python 3.9+**, relying on a robust ecosystem of open-source libraries to handle everything from massive data ingestion to deep semantic analysis.

Table S1 Software, Libraries, and Data Resources

Resource / Library	Role in Framework	Official URL / Source
Python	The core programming language that powers our entire analysis pipeline.	https://www.python.org/
Jupyter Notebook	The interactive environment where we prototyped ideas, explored the data, and generated our visualizations.	https://jupyter.org/
arXiv Dataset	Our primary source of data, accessed via the Kaggle/Cornell University OAI snapshot.	https://www.kaggle.com/datasets/Cornell-University/arxiv
GitHub Repository	The home for our custom code, making our full framework available to the community.	https://github.com/sidsharmaa/weak-signals-new
Pandas	The library we used for manipulating data tables, organizing time-series, and managing our datasets.	https://pandas.pydata.org/
NumPy	Essential for high-performance numerical calculations and handling complex array operations.	https://numpy.org/
Scikit-learn	We used this for calculating utility metrics, scaling data, and running the grid search to tune our parameters.	https://scikit-learn.org/
Apache Parquet (via PyArrow)	We chose this format for high-speed data storage. It allows for extremely fast reading	https://arrow.apache.org/

	and writing of large files, which kept our pipeline efficient.	
Pydantic	A validation tool we used to keep our data clean. It enforces strict schemas to filter out malformed records before they enter the pipeline.	https://docs.pydantic.dev/
spaCy	Our go-to tool for Natural Language Processing, handling tasks like robust tokenization and text cleaning.	https://spacy.io/
NLTK	The Natural Language Toolkit, which we used to remove stopwords and extract representative n-grams (phrases of one or two words).	https://www.nltk.org/
Sentence-Transformers	This library allowed us to generate deep semantic embeddings (using the all-MiniLM-L6-v2 model) to capture the actual meaning of keywords, not just their spelling.	https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
Matplotlib	The plotting library we used to create the static figures, including our Issue Maps, Emergence Maps, and Signal Matrices.	https://matplotlib.org/