Team Lead Name: Chris Milo

Team Lead Freelancer.com username: ChristopherTMilo

Submission Title: Automated Risk Information Extraction and Prediction for NASA Project Documents

Executive Summary

We propose a risk taxonomy and document processing pipeline to create tables of risk tuples (risk statement, likelihood, consequence, program name, date, topics, and other information) from arbitrary sets of project documents. The pipeline is developed by training language, image, and document formatting learning models on diverse corpora of NASA documents.

Extracting risks from existing project documents requires models that can represent and assign meaning to:

1. Sequences of text from knowledge domains common to NASA projects (e.g., STEM, government program management, government regulation)
2. Relative placement of text in 2D images (e.g., chart elements and table entries)
3. Document formatting (e.g., hierarchical outlines, bulleted lists, tables)

To identify and extract risk tuples from project documents, we propose using a set of binary classification models, trained by fine-tuning pretrained models (e.g. Bidirectional Transformers for Language Understanding, BERT[1]) on existing NASA risk tuples to predict the likelihood that a document element indicates a risk.

After a document element is tagged as likely indicating a risk, a new risk database entry should be created. To help populate the risk database fields for the newly created entry, we propose using the NASA project data (e.g., sample data provided by challenge organizers, lessons learned, and other documents such as those in the NASA Technical Reports Server) to train a series of deep learning models to extract and generate risk database field values. For example, a multi-label classification model should be used to help classify the risk category e.g. Cost, Schedule, Performance, and Technical. Regression models should be used to predict likelihood and consequence.

Proposed Risk Data to be Collected:

We propose collecting project risk data in two formats: (1) Risk Database and (2) Project Assessment Data. The following data structure is proposed for each format. An accompanying Excel workbook provides this structure along with anticipated data validation fields, where applicable.

[1] Devlin, Jacob et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." ArXiv abs/1810.04805 (2019)

Risk Database	Program Assessment
Date	Period
Year	Year
Project/Program	Project/Program
Project Description	Project Description
NASA Mission Directorate(s)	NASA Mission Directorate(s)
Organization	Organization
Project Phase	Project Phase
Risk ID	Technology
Risk Name/Title	Technology Readiness Level
Risk Description/Statement	Performance Area
Context	Severity
Status	Rationale
Status Date	Classification / Category
Status Description	
Mitigation Steps	
Open Date	
Planned Closure Date	
Likelihood	
Consequence	
Severity	
Criticality	
Affinity	
Approach	
Trend	
Classification / Category	

Legend:

Data fields indicating time of when the risk/assessment occurs
Metadata around program attributes to aide in risk classification
Data to be extracted and populated in template
Risk classifications performed (e.g., language models)

Table 1 Proposed project data formats

In both data sets, common metadata is to be collected for each instance of risk or assessment captured. Metadata fields include:

- Project/Program
- Project Description
- NASA Mission Directorate(s)
- Organization
- Project Phase (at point in time where risk or assessment occurs)

While the models proposed in the next section can be used to attempt extracting or abstracting this information from the documents, this will likely be a largely manual process of building the project/program attributes. The purpose of collecting project metadata is to enhance further classification of project data where like programs / systems can be identified and related to future programs. Those similarities can lend to further investigation of similar risks occurring.

Most of the risk data fields are to be captured via methods discussed in following sections. Human review of automatically generated risk database entries can in turn aide in improving the models by providing additional data for learning model training and clustering.

Risk Database

Our proposed risk database takes unstructured risk reporting from Quarterly/Mid-Year/Annual program review documents and structures data so that it can be ingested into predictive models. It not only captures pertinent program risk information snapshots, but also captures the reporting over time to better understand the evolution of risks. Additionally, the simplicity of capturing likelihood and consequence scores in a database form can yield analytical results as NASA tracks a particular risk's evolution over time and can further determine if mitigation strategies were successful.

Program Assessment Data

Program assessments contain higher level narratives around program performance that we believe should also be captured and assessed. These assessments are more free form narratives that are captured along with trends around severity in the areas of cost, schedule, technical, and programmatic.

Data Collection Process

While we describe methods and approaches to automate the capturing of historical risk data in subsequent sections, we also recognize the need for a comprehensive collection approach going forward. That is why we also propose the standardized templates for programs to capture all risks going forward. Toolsets can then be developed to present/visualize the risk information in commonly used formats such as risk cubes and risk slides. This will reduce duplication of data entry when programs build their program review slides and documentation while serving as a forcing function to capture risk data across the entire organization in a consistent manner. Programs can simply query the risk database and run scripts to present the information in a visually simplified manner.

Risk Classification / Categorization

The below figure is a notional example of potential risk classifications that can be made based on our analysis. As you further decompose risks into lower categories, programs can then use the classifications to predict risks for ongoing and future programs.

Progm/Other						Operational										Sch			Cost				Technical											
Organizational Issues	Assumptions	Service Strategic Vision	Strategic Vision	Acquisition Process	Advocacy	System Redundancies	Vulnerabilities	Training	Safety/Health	Availability	Sustainability	Reliability	Technical Obsolescence	Mission Effectiveness	Environment	Threats	Delays	Fielding Timeline	Acquisition Timeline	Contractor	Estimating Risk/Uncertainty	Funding	Budget	Standards/Protocols	Expandability	Certification/ Accreditation	Information Assurance	System Integration	System Dependencies	Interoperability	Performance Parameters	Production	Test and Evaluation	Software	Design	Requirements

Table 2 Notional risk classification scheme

Proposed Single Method to Populate Existing Risk Data and Predict Risks

We propose fine-tuning pretrained deep learning transformer models that can process text, images, and document formatting across a variety of NASA project document types to determine if risks are

indicated and if so, predict and populate the associated risk field values. A single document processing pipeline is used to parse text and graphics from a variety of documents and then:

1. Identify *risk indicators*, defined as aspects of a program that indicate the likely presence of a risk. These include, but are not limited to, existing *explicit risks*, defined as aspects of a program that have already been clearly identified as a project risk by document authors (often collected into dedicated risk documents with associated risk field values such as risk name, topic, likelihood, consequence etc.)
2. For each identified risk indicator, automatically populate values for most of the risk database fields

A single binary classification model outputs the probability (0-1) that a document element indicates a risk. *Explicit risks* are treated as a subset of *risk indicators*, essentially classified as very strong indicators. For example, consider the sample explicit risk written in Figure 1:

> *Risk #7 TRL Uncertainty: If the technology is not approved as TRL 3 or above, the program will have to further prove out viability delaying deployment by a year and requiring funding for 2 expert FTEs.*

The proposed document processing pipeline would classify the statement as a strong risk indicator with high probability (e.g., $p > 0.95$).

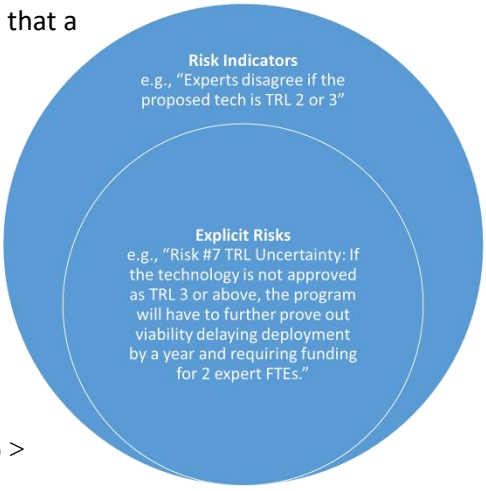
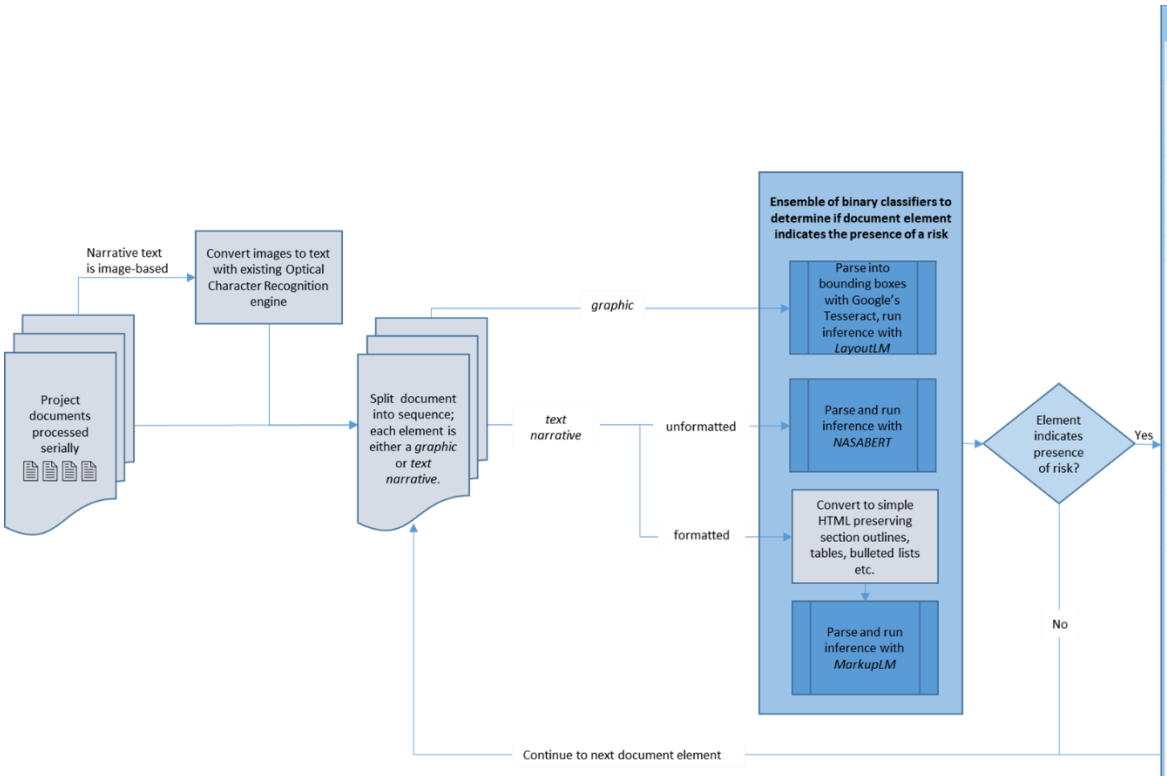


Figure 1 Risk indicators

This combined pipeline enables the use of a common framework to map both existing explicit risks and new (or *predicted*) risks to a common database. Sorting the database by risk indicator probability will then approximate a partition of the risks into sets of explicit risks and new risks. See Figure 2 for a summary of the document processing pipeline.

Figure 2 Document processing pipeline



Add entry to risk database. Automatically populate most field values.

Risk Database Field	Method to Populate Field Value
Date	Information extraction model
Year	Information extraction model
Project/Program	Information extraction model
Project Description	Information extraction model
NASA Mission Directorate(s)	Information extraction model
Organization	Information extraction model
Project Phase	Information extraction model
Risk ID	Assigned sequentially
Risk Name/Title	Concatenate Risk ID & classification
Risk Description/Statement	Sequence to sequence language model
Context	Populate with parsed Document element that fed the binary classification step
Status	Default to "open/unmitigated"
Status Date	Current date
Status Description	Default to "open/unmitigated"
Mitigation Steps	Human entry. Use text sequence similarity to suggest related lessons learned
Open Date	Current date
Planned Closure Date	Unpopulated
Likelihood	Regression model (inputs are language model text features, output is [0-1] float
Consequence	Regression model (inputs are language model text features, output is [0-1] float)
Severity	Function of likelihood and consequence
Criticality	Function of likelihood and consequence
Affinity	Human entry
Approach	Human entry
Trend	Human entry
Classification	Multiclass classification model

Training Data Generation

To transform the existing NASA information into the proposed format, we developed a document preprocessing algorithm to extract text data from NASA documentation such as the GCD documents provided for this Challenge. We used an Apache Tika framework to convert the different file types (docx, pptx, and PDF) into string text.

Preprocessed documents are parsed into sentences / paraphrases using the Natural Language Toolkit, and cleaned text sequences are outputted into CSV files for use in downstream classification tasks. Each text sequence is manually annotated with a Boolean marking those text sequences that have been judged as risk indicators[2]. This process can easily be repeated for a much larger corpus of NASA documentation, but the method has challenges parsing sentences for PowerPoint documents due to

[2] For the small data set available in this challenge, annotations are based on analyst assessment. Extending this to a large gold standard training data set would likely require a training data working group to systemically collect documents, establish rigorous annotation methods and schema, and finally review and annotate. A free open source tool for fast and efficient team annotation is doccano (https://github.com/doccano/doccano)

their unstructured nature. Additional research and analysis needs to be performed in this area and an evaluation of the feasibility of using PowerPoint for the purposes of the project. Images can be separated from narrative text using docx2txt, PyMUpdf, and PyPDF2 packages, and classified by type (e.g., risk cube, schedule, process diagram, system architecture, line chart etc.) by fine-tuning Google's pretrained Vision Transformer image classification model[3].

DistilBERT[4] is fine-tuned using text sequences parsed from NASA documents. The text sequences are manually labelled with a binary risk classification, risk affinity, likelihood, and consequence. The source code for the document preprocessing and training is available on the team's repository.[5] The following three subsections detail the models to be fine-tuned and used for both classifying document elements as risk indicators and populating the associated risk database field values.

Fine-tuning Transformers Models for Unformatted Text

The project team explored a "Zero-Shot Classification" method using a Hugging Face pipeline to predict the likelihood that a text sequence was associated with a candidate label "Risk," which would allow for an unsupervised and automated (but less robust) methodology to classify risks[6]. The "Zero-Shot Classification" model produced an accuracy of 73% based on the binary risk labels that were manually labeled. This method would allow an out of the box solution for classifying risk indicators and would not require manual risk categorization input; however, it is unlikely that it would accurately classify new risks.

To better classify risk indicators, we developed a transfer learning classification script to fine-tune pretrained language models, such as BERT[7]. The parsed sentences and binary risk labels were fed into the pretrained language models and used to predict on a testing data set. To account for the imbalanced nature of the data (only a minority of text sequences are risk indicators) the risk class weights were calculated and incorporated into the loss function when training the model. The accuracy of the transfer learning models was 97% on testing data that was held out of the training and validation data sets. Compared to the Zero-Shot Classification method it is a significant improvement, but examining the performance on the risk sentences only, the accuracy was around 50% - 67%, depending on the pretrained language model employed. This is primarily due to the imbalanced nature of the data and the small sample size – there are only 2,363 sentences / paraphrases and 5% are labelled risks. Adding more training data to the model would help the language model to better predict risk sentences.

The input data for the binary risk classification can be found in the Risk Data folder in the team's git repo. The python script labelled "risk_binary_class.py" performs both the Zero-Shot Classification and the transfer learning classification. The required python packages are listed in the "requirements.txt" document.

Once a risk has been identified it can be further classified into a Risk Affinity (Cost, Schedule, Technical, or Programmatic). A similar process to the binary classification was employed in which risk sentences

[3] This classification by type is a graphic pre-processing step to determine graphic relevancy before establishing and analyzing a single graphic's set of bounding boxes

[4] Pretrained model is hosted at https://huggingface.co/distilbert-base-uncased

[5] https://github.com/uwts/ProjectRisk (Github username is *uwts*, Github repository name is *ProjectRisk*)

[6] https://huggingface.co/docs/transformers/v4.16.2/en/main_classes/pipelines#transformers.ZeroShotClassificationPipeline

[7] Before fine-tuning, we propose extending an existing pretrained BERT model with NASA documents and other documents from relevant domains using methods detailed in exBERT: Tai, Wen-Hsin et al. "exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources." FINDINGS (2020).

with the labelled risk affinity were trained with a pretrained language model to predict the Risk Affinity on a test set of data. As with the binary classification, the class weights were used in computing the loss function in training to account for class imbalances. The accuracy of the Risk Affinity Classifiers ranged from 50%-55% on the testing data held out of training and validation.

The input data for the risk affinity classification can also be found in the Risk Data folder in the team's git repo. The python script labelled "risk_afinity_class.py" performs the transfer learning classification. The required python packages are listed in the "requirements.txt" document.

Fine-tuning Transformers Models for Graphics

We propose fine-tuning the LayoutLM Transformer model to classify elements of, and extract information from, document graphics (e.g., figures, charts, risk matrices). LayoutLM *jointly models interactions between text and layout information across scanned document images, which is beneficial for a great number of real-world document image understanding tasks such as information extraction from scanned documents*[8]. The pre-trained model, available on Hugging Face[9], can be fine-tuned by collecting and annotating relevant document graphics. Figure 3 contains a screenshot from our graphic data annotation process. Images are uploaded to the web tool[10] that runs OCR to extract text tokens and associated bounding boxes (pixel coordinates for text tokens). Sequences can then be manually classified and linked (see colored annotation on the left below) for exporting and fine-tuning. See the team's git for annotated data exports and sample train/test code.

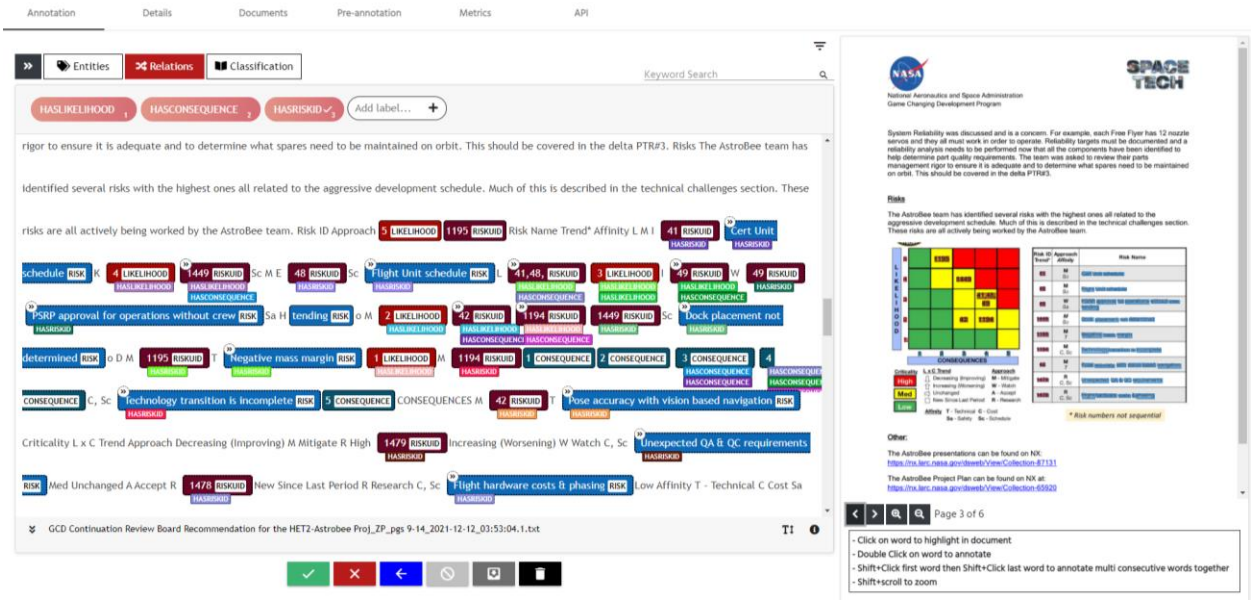


Figure 3 Annotating document graphics for fine-tuning LayoutLM

As shown above, the LayoutLM model is intended to both identify risk indicators and extract associated risk database field values to populate a new database entry (above we have linked risk names with likelihoods and consequences).

---

[8] Xu, Yiheng, et al. "LayoutLM: Pre-Training of Text and Layout for Document Image Understanding." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, July 2020. Crossref, https://doi.org/10.1145/3394486.3403172.

[9] https://huggingface.co/microsoft/layoutlm-base-uncased

[10] We used the UBIAI annotation web service

Fine-tuning Transformers Models for Document Formatting

We propose fine-tuning the recently-published MarkupLM Transformer model to classify elements of, and extract information from, formatted text narratives (e.g., hierarchical document outlines/sections, bulleted lists, tables). *MarkupLM was created for document understanding tasks with markup languages as the backbone such as HTML/XML-based documents, where text and markup information is jointly pre-trained. Experiment results show that the pre-trained MarkupLM significantly outperforms the existing strong baseline models on several document understanding tasks*[11]. The MarkupLM paper focuses on understanding and extracting data from structured website using the XPath information that can be pulled from the site's HTML string and embedded into the transformer model in parallel with the unstructured text. However, instead of focusing solely on websites, we propose a novel use of this powerful Document AI tool to better understand formatted project documents, by converting project documents (e.g., in docx or pdf format) into simple HTML files that preserve context-rich document formatting such as section headings, bulleted lists, and tables. This will augment the text sequence understanding power of BERT with formatting information that can be extremely useful to better comprehend a document.

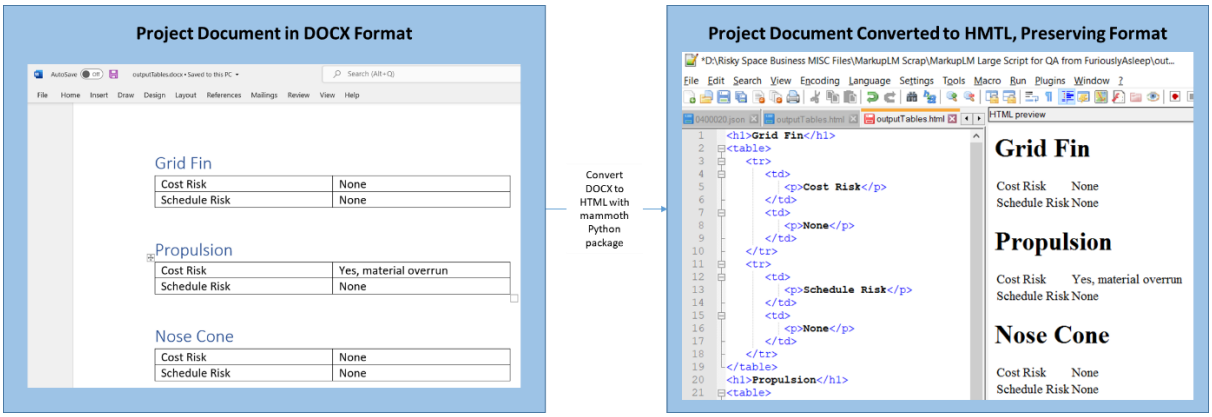


Figure 4 Preprocessing project documents for use in MarkupLM

We have made our fine-tuned model, trained on a small subset of the WebSRC dataset[12], available on Hugging Face[13]. Our team git contains a sample information extraction script that uses the fine-tuned model on a sample of a DOCX file that has been converted to simple HTML (see Figure 4) using the mammoth python package[14]. The MarkupLM model is new and has not yet been fully implemented into the Hugging Face transformer package. See instructions on our team's Hugging Face model card for a link to a branch of the transformers package that integrates the MarkupLM architecture.

Using Structured Risk Data, Lessons Learned, and Synthetic Data

Risks captured in the proposed risk database format and the publicly available lessons learned data can be used to improve the automatic generation of new risk tuples. Some examples:

1. The data can be used to further fine-tune the classification of risk indicators. The lessons learned fields of *Abstract*, *Lessons Learned*, *Recommendations*, *Driving Event* are rich with project narratives containing risk indicators.

[11] Xu, Yiheng et al. "LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding." ArXiv abs/2104.08836 (2021)
[12] Chen, Xingyu, et al. "WebSRC: A Dataset for Web-Based Structural Reading Comprehension." Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021. Crossref, https://doi.org/10.18653/v1/2021.emnlp-main.343.
[13] https://huggingface.co/FuriouslyAsleep/markuplm-large-finetuned-qa; recommend fine-tuning on formatted NASA documents
[14] https://pypi.org/project/mammoth/

2. There are 144 unique topics in the lessons learned that can inform the risk classification domains. They can also be paired with risk indicators identified in #1 above to predict labels from risk indicator narratives (e.g., Lesson learned #2596 has the following substring in the *Driving Event* field: "The X-38 schedule changed frequently due to the change in scope of the final spaceflight vehicle and ISS budget woes." This can be considered a risk indicator and associated with lesson's *topic* field value of "Program and Project Management" to start a multi-label classification training dataset that uses the risk indicator text to predict the risk topic / category.
3. The language models described above can be used to measure the semantic distance between a new risk indicator and each of the risk indicators in the lessons learned or the new risk database, returning the nearest neighbors and their associated risk data. For example, a close match to a risk indicator in the lessons learned database could prompt a human reviewer to reference the matched lesson's *recommendation* field value for relevant risk mitigation ideas to be applied to the new risk.
4. Regress embedded text sequences of explicit risks on likelihood and consequence (likely normalized to 0-1) to generate a model that can predict likelihood and consequence of new risks using the indicator text.
5. Pairs of *risk indicators* and associated *risk statement* can be used to train few-shot sequence-to-sequence language models that translate risk indicators into risk statements. E.g., "Experts disagree is the proposed tech is TRL 2 or 3" might be paired with "Risk #7 TRL Uncertainty: If the technology is not approved as TRL 3 or above, the program will have to further prove out viability delaying deployment by a year and requiring funding for 2 expert FTEs."[15]

Final Considerations and Summary

The models proposed in this paper are chosen because they are open source and, in many cases, implemented via user-friendly Python packages. The transformer based classifiers could be implemented separately, perhaps creating very similar or duplicative risk database entries. This would require additional human review but the task could be truncated to arbitrarily small review sets by applying high thresholds for classifying document elements as risk indicators. An ensemble of the models might attempt to combine the various architectures into one (e.g., jointly modeling text sequences, formatting, bound boxes, and graphics), or instead add an additional layer near the end of the network that takes output from each individual model. In any scenario, the final output will still be experimental, requiring careful human review and revision.

[15] We have deployed a simple publicly available web demo at https://riskgenrrun-sutjj3wk2q-uc.a.run.app/ that can take a very short txt file, identify text passages that are *risk indicators* using a semantic certainty based heuristic, and then output an attempt to translate the extracted risk indicators into new text sequences that are *risk statements*. The sequence to sequence component uses Google's T5, fine-tuned on our short list of synthesized (risk indicator, risk statement) pairs.