Text Mining Unveils Entities & Relations in Wikipedia's Aircraft and Nuclear Plant Data

2AMM30, Text Mining - (2023-GS1)

Group 5

Full Name	Student ID
Kaloyan Dragiev	1396048
Jelle Heinemans	1415131
Stefanos Karamperas	1797395
Victor Wen	1869698
Sidney Damen	1436864

Lecturer: Zeno van Cauter

Eindhoven, November 12, 2023

1 | Introduction

In this project, we delve into two distinct components. For Component 1, we select aircraft data as our subject matter. The initial phase of this component involves meticulous data annotation, a process in which we carefully label and categorize data elements. Subsequently, we construct a sophisticated model capable of extracting both entities and their relations from this annotated data.

In Component 2, we expand our horizons by applying the model developed in Component 1 to a more technical complex dataset associated with nuclear plants. Here, our objective is to extract entities and relations from this specialized domain. This transfer of our model from one context to another offers a unique opportunity to evaluate its adaptability and effectiveness.
The report is structured as follows:

- Exploration of the entities and relations of interest in both components.
- A thorough explanation of our data selection criteria that clarifies the rationale behind the selection of these particular datasets for analysis.
- An overview of the annotation process
- A detailed examination of our model development procedure that reveals the methods and strategies applied to produce a successful entity and relation extraction model.
- A thorough explanation and discussion of the findings and understandings obtained from using our models on datasets from nuclear power plants and airplanes.

We hope that this analysis will not only demonstrate our approach but also provide light on the complexities and difficulties involved in text mining in these particular fields.

2 | Identifying entities & relations of interest

In this section, we present the entities and relationships used in the labeling process. Detailed guidelines for these annotations can be found in the Appendix (Section D). For each entity an example is given between brackets. The relationships are in the format of {object_entity} relation_name {subject_entity}.

2.1 | Component 1

Entities	Relations
Category (Airliner)	{model} has_category {category}
County of origin (British)	{model} created_in {country of origin}
Date (1965)	{model} first_built_in {date}
	{model} first_flown_in {date}
Engine type (Turbofan)	{model} has_engine_type {engine type}
Era (60s)	{model} in_era {era}
Manufacturer (Boeing)	{model} created_by {manufacturer}
Model (727-100)	{[type X]} same_as {[type X]}
Number of engines (Single-engine)	{model} has_engines {number of engines}
Passenger number (100)	{model} has_passengers {number of passengers}
Wing type (117 ft)	{model} has_wing_type {wing type}

Table 2.1: Entities and Relations for component 1.

2.2 | Component 2

Entities	Relations
Event (Automatic reactor scram)	{[Type X]} same_as {[Type X]}
Date (2/18/87)	{Event} happened_on {Date}
Time (0001 hours)	{Event} happened_at {Time}
Cause (Personnel error)	{Event/Cause} caused_by {Cause}
Activity (Reduce generator load)	{Activity} has_participant {Participant}
Solution (Manual load reduction)	{Cause} fixed_by {Solution}
Component (Control rods)	
Participant (Operators)	
Location (Control Room)	{Event} occurred_at {Location}
System (Reactor protection system)	{System} has_part {Component}

Table 2.2: Entities and Relations for component 2.

3 | Data selection

3.1 | Component 1

3.1.1 | Extracting relevant data

We start the data filtration process by locating keywords that are specifically associated with aircraft, like "plane", "aircraft", and "airliner". After having the articles containing the keywords, we filter the data to only include the first section, i.e. up to the second header, because based on our observations, that's where the information is the most dense and the most relevant aircraft data is usually located.

During the annotation process we skipped any article that came up that was not directly about aircraft to make sure our training and validation sets are accurate. Using this practical method, we only choose articles that are thought to be pertinent to our goals. This methodology works well for creating targeted training and validation sets, but it might not work well for extracting comprehensive airplane data from the Wikipedia dataset as a whole. This approach's limitations are expounded upon in section 10.

3.1.2 | Set-up training and validation data

The articles in the validation set come from files *wiki_13*, *wiki_42* and *wiki_69* from both folder *AA* and folder *AB*. Furthermore we have the articles from file *wiki_63* from folder *AA*. These files were chosen at random.

3.2 | Component 2

3.2.1 | Set-up training and validation data

From the provided Excel file we exported the data under the column with the header 'abstract', as this contained information about the events in text format. The text contained some HTML tokens ('
'), so we did some preprocessing to remove those. After that we selected 200 random articles to be the validation data and the other articles minus those 200 to be the training data. Note that not all articles are labeled in the validation and training data.

4 | Annotation process

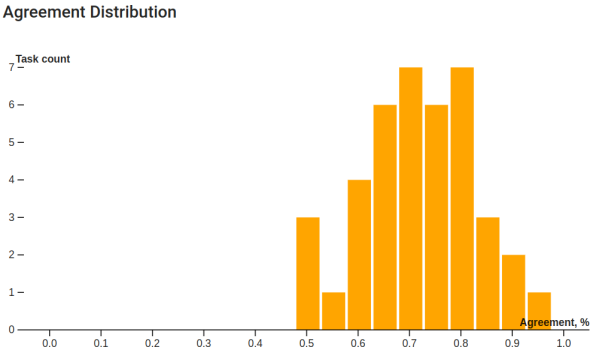
Name	Jelle	Kaloyan	Sidney	Stefanos	Victor
Jelle		76%	84%	81%	63%
Kaloyan	76%		77%	74%	61%
Sidney	84%	77%		74%	62%
Stefanos	81%	74%	74%		60%
Victor	63%	61%	62%	60%	

(a) Agreement Matrix - Component 1

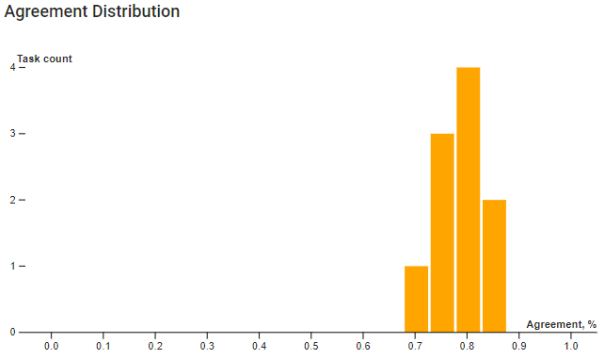
Name	Jelle	Kaloyan	Sidney	Stefanos	Victor
Jelle					
Kaloyan					
Sidney				79%	81%
Stefanos			79%		76%
Victor			81%	76%	

(b) Agreement Matrix - Component 2

Figure 4.1: Annotators' Agreement Overview



(a) Agreement distribution - Component 1



(b) Agreement distribution - Component 2

Figure 4.2: Agreement distributions

4.1 | Component 1

4.1.1 | Description

Initially, we selected 50 articles for our validation set, with each team member tasked to annotate 40 articles. This approach aimed to finetune the guidelines and achieve a robust agreement score, which typically falls within the 70th percentile range. Then, we discussed and agreed on the general labeling guidelines we would all follow while labelling our individually assigned articles and defining the relations within them, which are in Appendix D. Subsequently, we allocated 10 articles to each group member for creating a ground truth article, with each member assigned a package of 10 articles (the ground truth of other members) to review thoroughly for the validation set. For the training set, we annotated 250 articles.

4.1.2 | Evaluation

The process of labeling proved to be more time consuming than expected. Prior to defining our general labeling guidelines, our agreement percentage based on individual annotations was significantly lower compared to our final agreement score after we agreed on (and used) labeling guidelines. In Figure 4.1a the agreement score is presented for every person compared with every other. A distribution of our agreement score with regards to the number of articles is presented in Figure 4.2a. Our mean score for agreement based on every article for all annotators is **71.93%**

We believe our final agreement score is satisfactory, given the inherent difficulties associated with labeling aircraft data. We present a list of the main issues we encountered:

- There are a lot of ways to refer to the same aircraft model (e.g., Boeing 787, 787, 787-9, "Dreamliner" etc.)
- There is a significant number of aircraft categories (e.g., military transport) and roles associated with them (e.g., anti-submarine warfare), making it hard for the annotators the perfectly align on what specifically constitutes the role of a given aircraft model
- In our Wikipedia dataset, it is very common for aircraft articles to lack some of the entities we are interested in (a common example of thus is "engine type"). Thus, annotators were required to "improvise" in order to somehow extract information for frequently missing entities (e.g., given the reference of "jet airliner" within an aircraft's article, we can deduct that "jet" is the abstract type of engine the aircraft uses). This was not ideal in many cases, but did provide us with considerably more information we can use for the model training/benchmarking process.
- As with any text mining task, some understanding of the articles' context was required in order to properly label them. Although this did not prevent us from achieving what we believe is a more than satisfactory result in terms of labeling, it did slow down our progress and became one of the main contributing factors in cases were the agreement score of annotators was low.

We also suspect that some of the disagreement is due to spaces in the selected regions for entities. This is because selecting just the text without the spaces before and after it is cumbersome, and can slip by easily. Inconsistency in this decreases the agreement, while the actual entities can be still equal. Furthermore, Victor's scores are lower than the rest of the group. This discrepancy can be attributed to the fact that he conducted a considerable amount of annotation work prior to the establishment of the main guidelines, and subsequently did not revise or align his annotations with the finalized guidelines. It's worth noting that annotations made after the formulation of the main guidelines demonstrated consistency with the overall group standards.

4.2 | Component 2

4.2.1 | Description

For the validation data of component 2 we decided to do 3 annotators instead of the minimum of 2, as most of the articles had elements that could be interpreted differently and contained ambiguity. Initially, we annotated five identical articles to assess the degree of overlap in annotations. Subsequently, we engaged in a collaborative discussion to establish comprehensive labeling guidelines for this dataset, which are detailed in Appendix D.

For the validation set, we carefully annotated 30 articles. Each team member was responsible for creating a ground truth article, with 10 articles allocated to each member for in-depth review and analysis. For the training set, we annotated 80 articles in total.

4.2.2 | Evaluation

The annotating for component 2 proved to be more difficult than the annotating of component 1, as the domain was more tedious and the lack of knowledge in this domain gave some difficulties as to what information is relevant enough to be annotated. After annotating a few articles with the standard labels and relations provided in the assignment description, we decided these were not sufficient enough and contained some ambiguity. Therefore we decided to change them to the ones that are currently in subsection C.4 and subsection C.5.

After annotating the validation data with the final labels and relationships we reached an average agreement score of **79.88%**, and the intra-annotater scores can be found in Figure 4.1b and the agreement distribution in Figure 4.2b. From this matrix it can concluded that for component 2 we have a similar matrix compared to component 1. Furthermore, from the distribution it is clear that the ranges in scores for component 2 varies less than in component 1, meaning the annotating is more consistent. Some of the most common issues are discussed below:

- Most of the discrepancies came from the variation of some of the labels. For example, Date, Time and Location were almost always in the same format, making them easy to label, but for Event, Cause, Attribute and Solution this was often not the case. Although Cause and Solution were often preceded by combinations of words giving a heads up of what was coming such as 'caused by' and 'due to' to state that the following piece of text was a causation and combinations such as 'corrective actions are/were', '... was replaced' often introduced the Solution label. This was unfortunately not as clear for the Event or Activity labels. We often agreed on when something was an Event or an Activity, but we often disagreed on where the end of the label should be as these labels often included information that one considered important but the other not important.
- Another often occurring disagreement came from the 'same_as' relation. Although we discussed beforehand when to use this relation, we were a bit inconsistent in actually using the label. Often this was the result of a label referring to another label too far back and using the relation because of the distance between them or simply not wanting to overdo the relation to avoid an abundance of relations.

5 | Task modeling

5.1 | Evaluation Dimensions of Knowledge Graph Quality

In evaluating the quality of our knowledge graph we have chosen to focus on *Accuracy*. Emphasising accuracy is crucial since it ensures the correct representation of entities and relationships. Additionally, choosing accuracy aligns with the optimization strategy of most NLP models.

Moreover, we splitted our accuracy scores for tokens that are not named entities/relations and those that are (empty label accuracy and non-empty label accuracy). This will give us more transparency for what our model is doing, since achieving higher accuracy can be deceiving and not totally correct (e.g. labeling everything as an empty label).

5.2 | Knowledge Extraction

Knowledge extraction was done by using the two-way approach. Using a two-way approach for Knowledge Extraction, involving separate steps for Entity Recognition and Relation Recognition with Error Propagation, offers several advantages over a single-model approach. Firstly, it allows for modularization - for each task we use differently fine-tuned model. This facilitates model updates or replacements without disrupting the entire pipeline. Additionally, the two-way approach brings more transparency, allowing for better understanding and analysing errors at each stage independently. Lastly, this approach often results in more efficient training, as models can be adapted to the specific complexities of entity and relation recognition, leading to improved overall performance. Here is what our different model expect and return as input and output:

Entity Recognition model:

- Input: A randomly selected article in plain text.

- Output: A comprehensive list of entities corresponding to each word in the article.

Relation Recognition model:

- Input: A randomly selected article in plain text and the list of entities obtained from the Entity Recognizer.
- Output: A systematically labeled list of all possible relations, indicating the connections between entities identified in the input text.

6 | Model selection

The model we selected for component 1 and 2 was BERT (Bidirectional Encoder Representations from Transformers). Considering our fairly limited compute resources, choosing BERT over models like GPT, LLM, RoBERTa, or others comes by its lower computational demands for smaller pre-trained models ("bert-tiny" or "bert-small"). BERT's bidirectional context understanding and effective transfer learning can be also advantageous for tasks like entity/relation recognition, even with a smaller amount of training data (like the one we have in this project). The specific implementations of the smaller Bert models were taken from Huggingface user prajjwal1.

For entity and relation recognition the model was implemented using the *BERTForTokenClassification* and *BertForSequenceClassification* architectures, respectively. Both architectures are part of the Transformers library provided by Hugging Face. Additionally, for word tokenizer we have chosen the *BERTTokenizer* - used for both architectures. It handles the preprocessing of text, ensuring compatibility with BERT's input requirements.

6.1 | Challenges

Along the process of implementation of the models we have stumbled upon several obstacles. Firstly, we had a problem that our Entity recognizer model always predicted every word as a non-labeled one. Considering that, our article text contains of very small portion of labeled text, labeling everything as non-label gets the model a very high accuracy already (without learning anything). We approach the problem by implementing a custom loss function that penalizes the model if the predicted label was a non-labeled one. However, we have found our after running several tests that the issue was due to our poor choice of learning rate for our model. It was too big and the function kept jumping around the local minima and that's why our results were not as expected.

Furthermore, another issue that arose during implementation was that our Relation recognizer model was using too much GPU memory and always crashed our python instance. At first we identified that it was due to a memory leak that we fixed in the code by changing were we store and call our model outputs. However, the problem still persisted after that change and we figured that it was caused by our ambitious choice of Bert model - Bert-cased. Considering that we only have a fairly limited number of compute resources, Bert-cased was even unable to complete a single epoch without reaching the GPU memory limit. Therefore, we opted to use less memory demanded Bert pre-trained models like Bert tiny, mini and small.

7 | Experiments setup

In order to find the best hyperparameter values for our models, we used a tool called Weights and Biases [1] (wandb) to carry out a grid search. All logged metrics can be found online here.

The specific options considered in the grid search differ from model to model. A minimum set of considered hyperparameters is listed in Table 7.1. The specific settings used can be found in the wandb projects as well, by selecting a project and navigating to the "Sweeps" tab.

Hyperparameter	Values considered
Learning rate	1e-5, 5e-5, 1e-4, 5e-4
Batch size	20
Epochs	20
Model size	tiny, mini, small

Table 7.1: Basic hyperparameter values used in grid search

8 | Results

8.1 | Component 1

The accuracies of the best runs for all 3 sizes of Bert we used can be found in Figure 8.1. We can clearly see here that there is a big jump in accuracy when going from Bert-tiny to Bert-mini, and a smaller jump when going to the even larger Bert-small. It is also interesting to note that the optimal value for the learning rate is about an order of magnitude higher for the relation recognition, when compared to the entity recognition.
The extended accuracies, thus with the empty and non-empty label accuracies, for each of these models can be found in Table 8.1.

Model	Accuracy entities (non-empty - empty)	Accuracy relations (non-empty - empty)
Bert-tiny	85.2% (61.6% - 94.6%)	69.0% (14.8% - 96.9%)
Bert-mini	89.7% (74.1% - 95.5%)	85.4% (70.9% - 92.9%)
Bert-small	92.4% (80.8% - 96.3%)	88.2% (79.3% - 92.8%)

Table 8.1: Best accuracies per model for component 1

Figure 8.1: Accuracies for the different models on the component 1 dataset

8.2 | Component 2

Similar to component 1, the accuracies for the best runs are plotted in Figure 8.2. Here again we see a big jump in accuracy when using the Bert-mini model, but now we see another such big jump when going to Bert-small. We can also again see that the optimal learning rate differs by about an order of magnitude between the entity and the relation recognition.
The extended accuracies for each of these models can be found in Table 8.2.

Model	Accuracy entities (non-empty - empty)	Accuracy relations (non-empty - empty)
Bert-tiny	59.1% (34.8% - 83.1%)	68.8% (20.2% - 93.1%)
Bert-mini	65.1% (44.7% - 84.3%)	80.2% (51.7% - 94.5%)
Bert-small	70.7% (57.6% - 82.8%)	86.1% (69.7% - 94.3%)

Table 8.2: Best accuracies per model for component 2

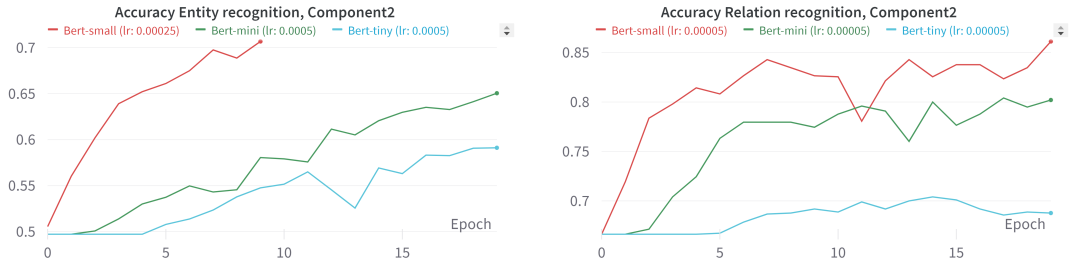


Figure 8.2: Accuracies for the different models on the component 2 dataset

8.3 | Knowledge graph & Attention

The knowledge graph visualization consists of all predicted labels that have a relation between them. Each different entity is visualized with separate color. The arrows show from where the relation is coming (e.g. *T-211* plane model has_category to *light* category). An example of our knowledge graph for a random article can be seen on Figure 8.3.

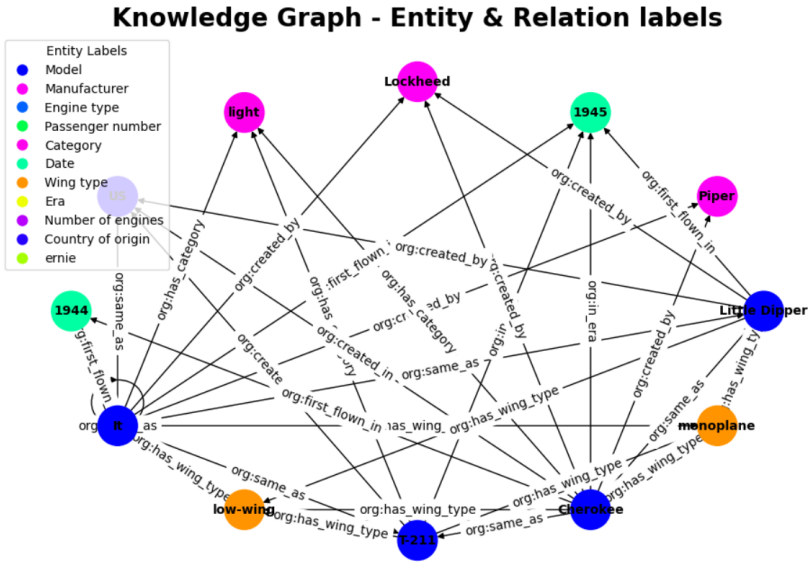


Figure 8.3: Knowledge graph for a random article

Furthermore, we have visualized all entity and relation attention weights using a tool called BertViz [2]. It provided us with a way to view all attentions per layer and head for each relation and entity model. This gave us light on what our model is doing on what stage of the model and whether it is paying attention to the important words in the articles. An example of our attention visualization for entity recognizer model can be found on Figure 8.4

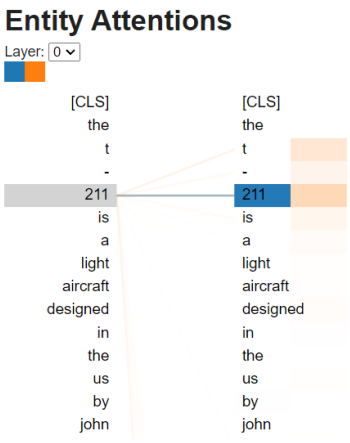


Figure 8.4: Entity Attention Visualization per Layer and Head

9 | Conclusion

In Component 1, we observed that a larger model generally improves results, but the extent of improvement diminishes as the model size increases. This suggests a trade-off, where a bigger model might yield better results, but it could also risk overfitting and performing worse. Considering our inter-annotator agreement of 80%, achieving 90% accuracy in Component 1 is very good.

Moving on to Component 2, we noted significant overfitting during training, with the model almost reaching a perfect 100% accuracy. Despite this overfitting, increasing the model size consistently improved results. The fact that in component 2 the relation recognizing model outperforms the entity recognizer is not surprising. During annotating the training and validation set, the annotators found that they could often label a word with multiple labels. Since the model only supports predicting a single label per token, this makes it hard for the model to learn what label to choose in these cases. Overall, given the smaller training set and more complex data, an accuracy of 70% to 86% is higher than expected.

10 | Discussion

Although the presented model achieves a decent accuracy score, it is clear there are some shortcomings in its design. The most obvious shortcoming is that the model is only able to predict a single label to each entity or relation. In most cases this is sufficient, but not always. Consider for instance the following sentence:

> The twinjet is powered by General Electric GEnx or Rolls-Royce Trent 1000 high-bypass turbofans.

Here the word "twinjet" is used to talk about the airplane, so it would be a "model", which gets a "same_as" relation to a previous mention of the specific airplane model. However, "twinjet" also relates to the fact that the plane has 2 engines, and they are of the jet type. This entity can thus be labeled in 3 ways, and indeed should be labelled all ways to extract all the information in the text. The presented model however only assigns a single label, in this case "model", causing it to miss some information.
It is possible to create a BERT based model that can predict multiple labels per token. This can be done by replacing the softmax activation function on the output with a sigmoid activation, since this activation functions allows multiple output classes to have a high value. Then the target labels simply include all of the appropriate labels for a token, and the model learns to predict multiple labels if necessary. However, since occurrences of this complication are rare in

the airplane data, it is difficult to create a training set with enough of these cases so that the model can learn this.
In the reactor incident data this is a more common occurrence, making it possible to create a sufficiently large training set containing instances where multiple labels are given to a token. However, due to time constraints this was not implemented in the model.

Another shortcoming of the model is that if the first part of the model does not properly recognize entities, the second part can not properly extract the relations. There are two ways in which this can go wrong. Firstly, if the entity recognizer does not think a certain token is an entity of interest at all, it will obviously never be considered to be in a relation whatsoever, causing the model to miss this information. On the other hand, if the entity is recognized as relevant, but wrongly classified, i.e. as a manufacturer instead of a model, the possible relations that entity can have with other entities are different from what the model thinks. To mitigate that the model could be modified to always predict the relation between entities, even if those entities should not be able to have a relation due to their type. However, this massively increases the processing time of inference, and introduces a big possibility for false positives.

A shortcoming in the data selection part of the model, is that the only filtering on the wikipedia articles is filtering by keywords. This is good enough to make selecting training and validation data easy, but not for extracting all airplane data from the whole dump. An improvement in this regard would be implementing some form of semantic search, which evaluates each article and classifies it as airplane-related or not. The model needed to do this is relatively simple, as it is closely related to the relation extraction model. An example way to implement this semantic search is to create a dataset with positive and negative examples from the whole wikipedia dump, and train a *BertForSequenceClassification* model on it. Then to extract all data from all of the articles, simply run each of the articles through this model and discard the ones that it deems not relevant. This was again not implemented due to time constraints.

A | References

[1] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[2] Jesse Vig. Bertviz - visualizing bert's attention. Software available from github.com/jessevig/bertviz.

B | Appendix: Contribution table

Student Name	Contribution (%)	Participation in tasks
Kaloyan Dragiev	20%	Took the lead on developing the text mining model and assisted in labeling
Jelle Heinemans	20%	Took the lead on developing the text mining model and assisted in labeling
Stefanos Karamperas	20%	Took the lead on labeling the data and assisted in the development of the text mining model
Sidney Damen	20%	Took the lead on labeling the data and assisted in the development of the text mining model
Victor Wen	20%	Took the lead on labeling the data and assisted in the development of the text mining model

Table B.1: Contribution table

C | Appendix: Descriptions

C.1 | Component 1: Entity Description

Category defines the role (use) of an aircraft
Country of origin is the country where the aircraft is originated from
Date is described in years and can be used multiple times in the relations.
Engine type is the type of engine used in an aircraft.
Era refers to multiple years or eras, such as the 1930s, or the World War II
Manufacturer is the company that produces an aircraft.
Model is the main type of an aircraft
Number of engines is the amount of engines an aircraft has.
Number of passengers is the number of seats (including the pilot(s)) a manufacturer specifies for an aircraft's factory configuration.
Wing type is the type of an aircraft's wing or how many wings an aircraft has

C.2 | Component 1: Relationship Description

created_by describes which manufacturer produces an aircraft
created_in describes in which country an aircraft is built
first_flown_in describes the year when an aircraft model carried out its first flight
first_built_in describes the year when the first aircraft of a specific model rolled out of the manufacturer's production line
has_category describes the role (use) of an aircraft
has_engine_type describes what type of engine is used by an aircraft
has_engines describes the amount of engines used by an aircraft
has_passengers describes the capacity of an aircraft in terms of passengers (according to the manufacturer's factory specifications).
has_wing_type describes the type of a wing of an aircraft
in_era describes in which era a model was used
same_as describes that an entity has the same type as another one (referring to the same entity)

C.3 | Component 1: Aircraft Categories Description

List of the most common aircraft categories provided to the annotators:

- airliner
- strategic bomber
- short-takeoff-and-landing
- utility
- prototype
- all-weather interception
- stealth interceptor
- heavy fighter
- fighter-interceptor
- short-range interceptor
- coastal patrol
- torpedo bomber
- strategic reconnaissance
- tactical bomber
- fighter
- mailplane
- airborne missile platform
- torpedo and reconnaissance bomber
- kit
- kit-built
- business
- flight test
- airborne ground surveillance
- airborne battlefield and group surveillance
- battle management
- command and control
- aerobatic trainer
- low-level attack
- ground strafing
- ground-attack
- naval fighter
- jet fighter
- personal
- Turbojet fighter
- multirole fighter
- stealth multirole fighter
- homebuilt
- maritime patrol
- light model
- bomber
- jetliner
- aerial refueling tanker
- passenger
- military transport
- ship-borne fighter
- sports and touring
- passenger transport
- search and rescue
- troop-transport
- airborne early warning
- liaison
- racing
- army co-operation
- regional airliner
- regional jet airliner
- regional jet
- corporate transport
- cargo
- day fighter
- agricultural
- cargo glider
- touring
- light passenger
- light reconnaissance
- light sports aircraft
- special light sports aircraft
- general purpose
- counter-insurgency
- light attack
- reconnaissance
- general aviation
- light fighter
- trainer
- transport
- demonstrator
- commuter
- observation
- freighter
- heavy bomber
- experimental
- high-altitude reconnaissance
- vertical take-off and landing
- medium bomber
- Flight training
- training
- glider
- light
- flying-boat
- convertiplane
- very long range fighter
- escort fighter

C.4 | Component 2: Entity Description

Event indicates something that happened, expected or unexpected
Date states the date on which something happened
Time states the time at which something happened
Cause refers to the what caused an event to occur
Activity is an action performed by humans, such as the plant operators
Solution refers to the action that was taken to fix the cause of the problem
Component is a part of the system, and refers to smaller objects within the system, such as a turbine or engine for example
Participant is the term for humans that perform actions in the nuclear reactor
Location refers to the location where something is located or happened
System indicates what system the problem is about

C.5 | Component 2: Relationship Description

same_as describes that an entity has the same type as another one (referring to the same entity)
happened_on indicates on what date an event happened
happened_at indicates at what time an event happened
caused_by states what the cause of an event is
has_participant describes which participant are responsible for which activity
fixed_by links the solution that was proposed to the cause of the problems
occurred_at states at which location the event occurred
has_part states parts that belong to the system

D | Appendix: Annotation guidelines

D.1 | Component 1

The main guidelines we followed were:

1. When labeling the "Number of Passengers" entities, only use numbers (e.g., for the phrase "78 passengers", only label "78"). However, for phrases such as "two-seat", or passenger number ranges (e.g., "120-180"), label the whole phrase.

2. When using the "same_as" relation we have defined, always draw the relation from the "identical" entity (source) towards the "base" entity (destination)

3. Always form a relation with the "destination" entity closest to the "source" entity (e.g., same sentence is prioritized)

4. Pronouns (e.g., "it") should be considered entities (and labeled as such) and connected to the "base" entity they refer to (e.g., Boeing 737) using a "same_as" relation

5. Similarly to the previous one, with phrases such as "The type" or "The plane" should be labeled as entities (including "the") and connected to the "base" entity they refer to using a "same_as" relation

6. The "same_as" relation we are utilizing can be applied to any type of entity (i.e., not just to the aircraft "model" entity where it is most commonly encountered)

7. When both an entity's abbreviation and full-text are provided (e.g., Defense Advanced Research Projects Agency, DARPA), proceed to label both entities and create a "same_as"

relation from the more abstract (abbreviation) to the more specific entity (full-text). Then, proceed to use the full-text entity for any other relations in the text.

8. Treat aircraft sub-models the same as their "base" models (e.g., 787-8 will be treated as 787, using a "same_as" relation)

9. Official aircraft nicknames should be treated the same as their "base" model entities (e.g., 787 "Dreamliner", a "same_as" relation is created from Dreamliner to 787)

Furthermore, to avoid confusion/conflicts during the process of labelling **aircraft categories** (roles), a list of the most commonly encountered categories was created. The list is provided in Appendix C.3.

D.2 | Component 2

The main guidelines we followed were:

1. When using the "same_as" relation we have defined, always draw the relation from the "identical" entity (source) towards the "base" entity (destination)

2. Always form a relation with the "destination" entity closest to the "source" entity (e.g., same sentence is prioritized)

3. Pronouns (e.g., "it") should be considered entities (and labeled as such) and connected to the "base" entity they refer to (e.g., "loss of offsite power") using a "same_as" relation

4. Similarly to the previous one, with phrases such as "The event" or "The activity" should be labeled as entities (including "the") and connected to the "base" entity they refer to using a "same_as" relation

5. When both an entity's abbreviation and full-text are provided (e.g., Loss of Power, LOP), proceed to label both entities and create a "same_as" relation from the more abstract (abbreviation) to the more specific entity (full-text). Then, proceed to use the full-text entity for any other relations in the text.

6. Anything with the word system, unit, or plant in it is a system (e.g. Auxiliary Feedwater System) is labeled as a system. These systems are also often noticable by their capital letters at the start of each word. Other parts that could be part of the system but we are unsure of are labeled as component rather than system. Furthermore we refer to the first occurrence of a system in the article with a "same_as" relation.

7. A "caused_by" relation can both go from event to cause, as from cause to cause, as it occurred that a cause is the consequence of another cause.

8. Location refers only to places (where the reactor plant is located), not rooms within the reactor plant e.g. operating room.

9. For most labels start labeling at the beginning of the sentence or after an article ('the' or 'a').

10. Only failures, failsafes (such as automatic scrams) or catastrophic occurrences are labeled as events, and must not be caused by humans: for that we use the Activity label.

11. To avoid confusion, always choose the most general form of an entity (e.g. 120V High Power circuit → circuit is labeled, or Group C control rods → control rods).

12. The Time label includes the word 'hours' or timezone (e.g. EST) in it. For Date, we include label day, month and year (not in this particular order: 7 November 2008 is labeled as one entity, just like 7/11/2008).

13. Activities must be direct actions (e.g. 'repairing', 'replacing', 'turning on') and not indirect (e.g. 'observing', 'determining') as these do not have direct results.

14. Plain years (e.g. '1957') are not labeled as Date to avoid confusing the the Time label.