# **Contact information for Official Representative:**

Name: Jim Liew

Email: [jim@sokat.co](mailto:jim@sokat.co)

Team Name: Team SoKat

# **Names of additional team members:**

Name: Li Sun, email: [lsun@sokat.co](mailto:lsun@sokat.co)

Name: Avinash Sharma, email: [asharma@sokat.co](mailto:asharma@sokat.co)

Name: Steven Tan, email: [stan@sokat.co](mailto:stan@sokat.co)

# **Introduction to Team:**

Our team consists of curious AI/ML software engineers and data scientists from SoKat Consulting, LLC. We are a Maryland based 8(a) WOSB company with strong ties to Johns Hopkins University and the local DMV area communities.

# **Executive Summary of Solution:**

Our solution combines a simple drag-and-drop UX/UI frontend interface for EULAs. The solution parses the EULA and predicts the probability that a given EULA clause is acceptable or not acceptable. Additionally, our solution allows the user to disagree with the prediction and send data back to retrain the AI models. The human-in-the-loop provides a mechanism to capture more labeled data and thus have a true continuous-learning and continuous-integration (CL/CI) deployment. We allow batch processing of documents and an easy one-page layout to evaluate and assess each clause, one at a time.

At any time, users can save their progress and revisit results at a later time. With a click of a button, results can be emailed to oneself or downloaded locally to their systems. To provide explainability, our solution uses attention algorithms to highlight key words that have been driving the AI model’s decision. Back-end processing adheres to best practices with regards to data processing and AI modeling.

Extensive time and effort were put into understanding the dataset as well. Data augmentation techniques were employed to expand the dataset to better train the AI models. Ensemble techniques were used to encapsulate the best of both realms of, traditional and deep neural network algorithms. Transfer learning was used at several stages in order to tweak and customize the AI models for our specific task. Immense thought was given while choosing the right AI models to ensure a logical flow of development.

# **SoKat Team’s Architecture:**

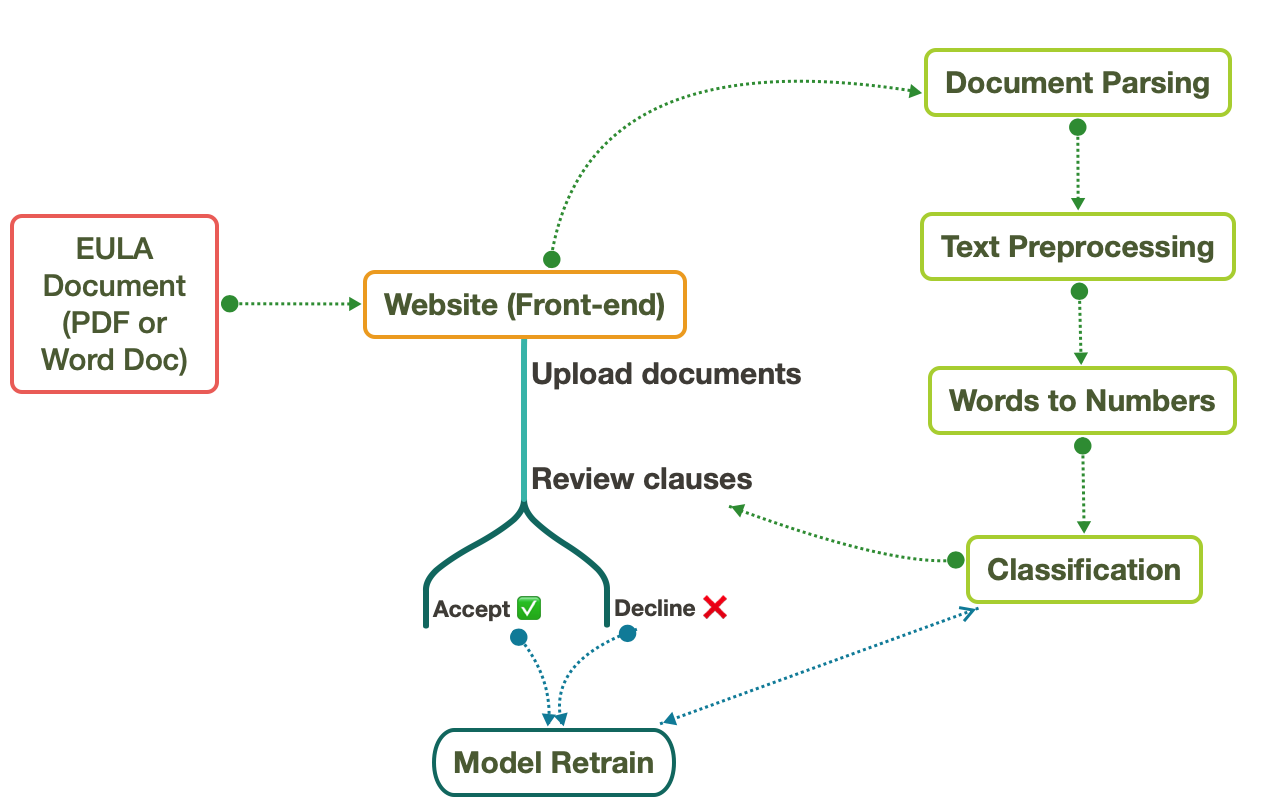


Figure 1: Overview of our solution presented as a flowchart. Front-end is where the user uploads EULA documents and reviews the clauses. Each clause is labelled as acceptable or unacceptable by the AI models in the back-end. The back-end engine components are shown in green boxes. The user can override the model’s decision and retrain it.

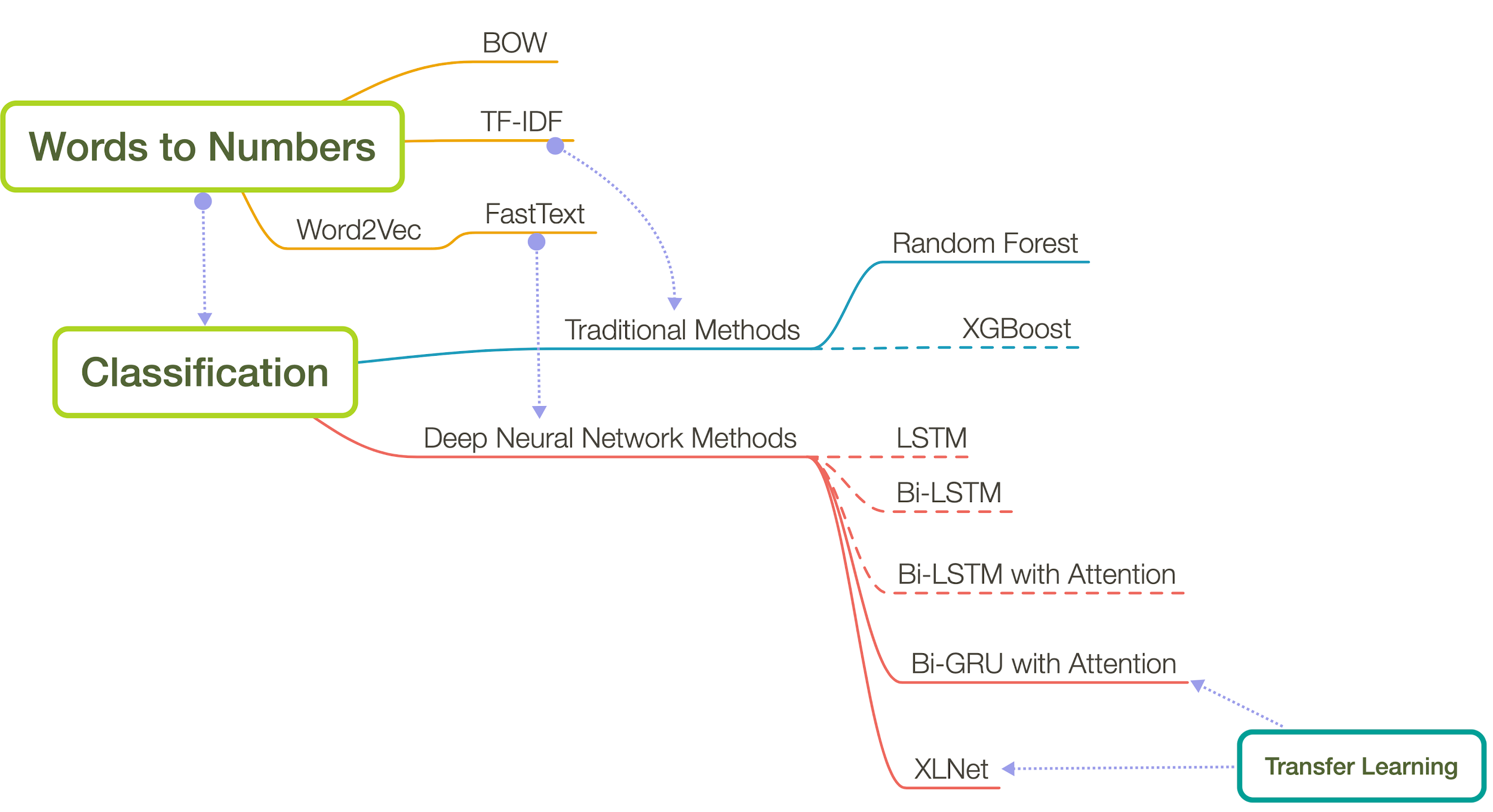
## **Technology Scope:**

* Front-end: ReactJS for web development.
* Hosting: Amazon’s AWS
* Back-end: Based on Python programming language
  + Document Parser: PDFMiner and PythonDocx (Python libraries)
  + Text Preprocessor: TextHero (Python Library)
  + Words to Numbers: Scikit Learn and FastText
  + Classification: Scikit Learn, Keras and TensorFlow

## **Functionality and User Interface:**

* What type of user interface does the solution provide?
  + We provide a ***web-based interface*** that allows users to access the EULA application through a computer, tablet or smartphone.
* What input formats does the solution support? (e.g. PDF or MS Word).
  + ***Both PDF and MS Word*** documents are supported by our application.
* How does the solution process batches of documents?
  + ***We allow batch processing*** of documents and enable users to assess each document one after another within a single page. Simply select multiple documents and drop it in the application.
* Can the user override the model?
  + ***Yes***, the user can decide to *agree or disagree* with the model’s suggestions. The model will take into account the user’s feedback and retrain overnight.
* At what level is the classification made?
  + Our application makes decisions at a **clause-by-clause** level. Furthermore, using advanced machine learning techniques of attention, we highlight words that are driving the model to make such a decision.

## **Application of Artificial Intelligence/Machine Learning (AI/ML):**

* **Provide a description of the ways in which the technology leverages AI/ML. Please specify general approaches (supervised, unsupervised) and conceptual description of how these apply to the challenge.**
* Our solution heavily relies on state-of-art AI/ML algorithms to evaluate the EULA documents. The flowchart illustrates the various supervised learning algorithms our team tried and tested during the process of this competition.   
    
    
  Figure 2: The expanded version of the back-end prediction engine. It comprises two major components: Words to Numbers and Classification. Arrows show which words to numbers algorithm used for which classification algorithm. Solid lines represent the models that were integrated into the final solution, while dashed lines represent the ones that we tested during the development process of our final solution.
* Word to Numbers: This section deals with the conversion of EULA texts to numbers that can be read and interpreted by the subsequent classification algorithms. For this step, we tested the following:
  + Bag of Words (BOW): This algorithm uses frequency of a word in the dataset as its identifier. Sentences are converted into a string of numbers, each corresponding to the frequency of the word in the text.

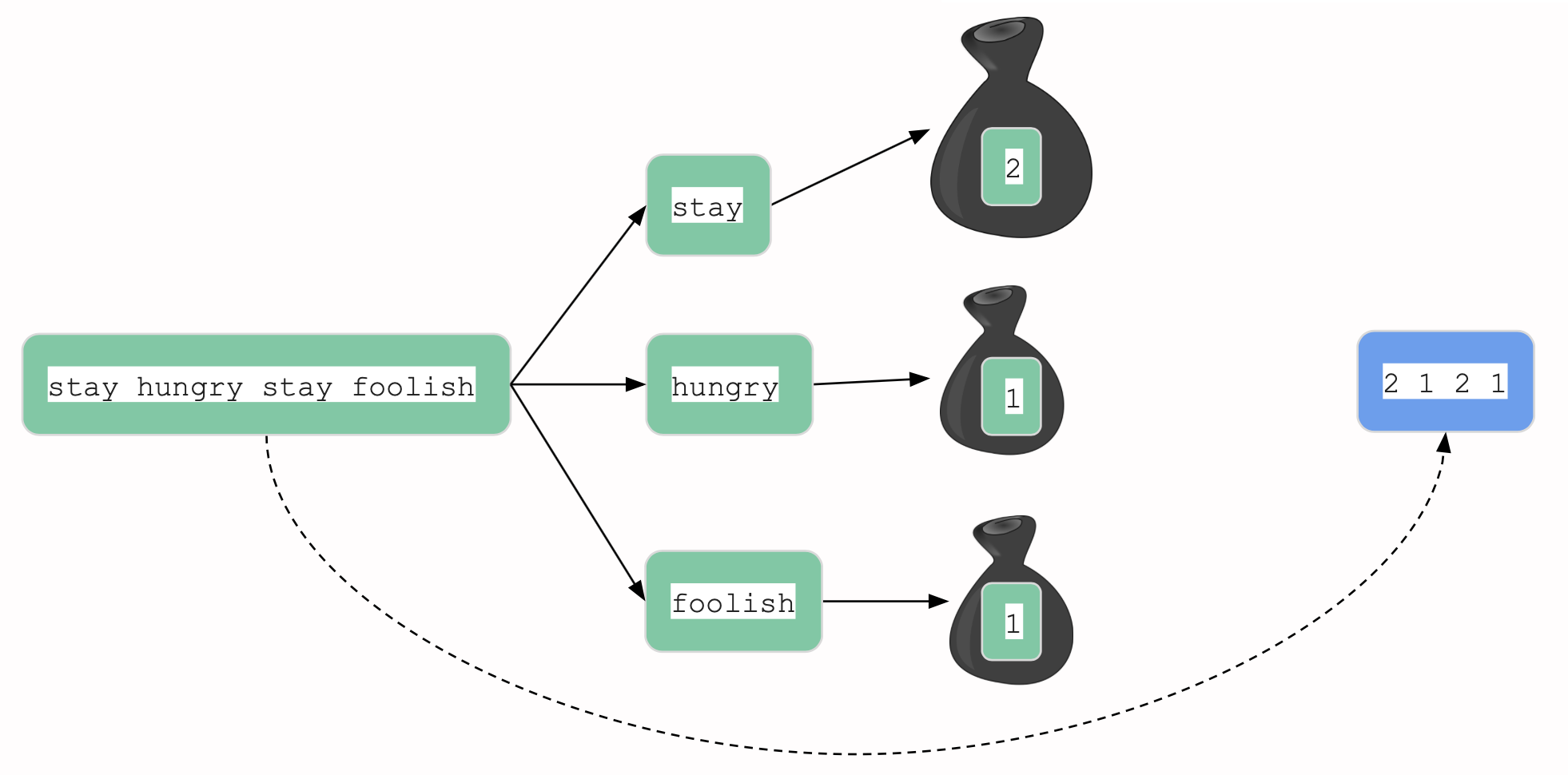


Figure 3: A pictorial representation of the BOW algorithm. Here the assumption is that the entire dataset contains just a single sentence: “stay hungry stay foolish”. A ‘bag’ corresponds to each of the words in the sentence with the value inside representing the frequency of the word in the sentence.

* + Term Frequency-Inverse Frequency (TF-IDF): This method takes a step above BOW by thinking about words in 2 dimensions:   
    1) frequency of word within a document, 2) frequency of word across all documents in the corpus (dataset).

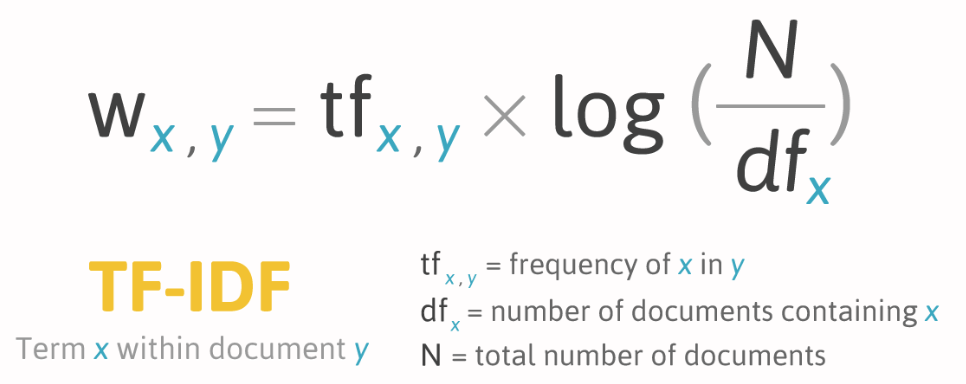


Figure 4: The mathematical equation that is used to convert a word (x) that is in a document (y) into a number.

* + [Word2Vec](https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf): This algorithm introduces the concept of “context”. It tries to represent words with similar vectors (or numbers) that are used in the same context. That way, similarly used words share the same mathematical representation - in this case vectors.

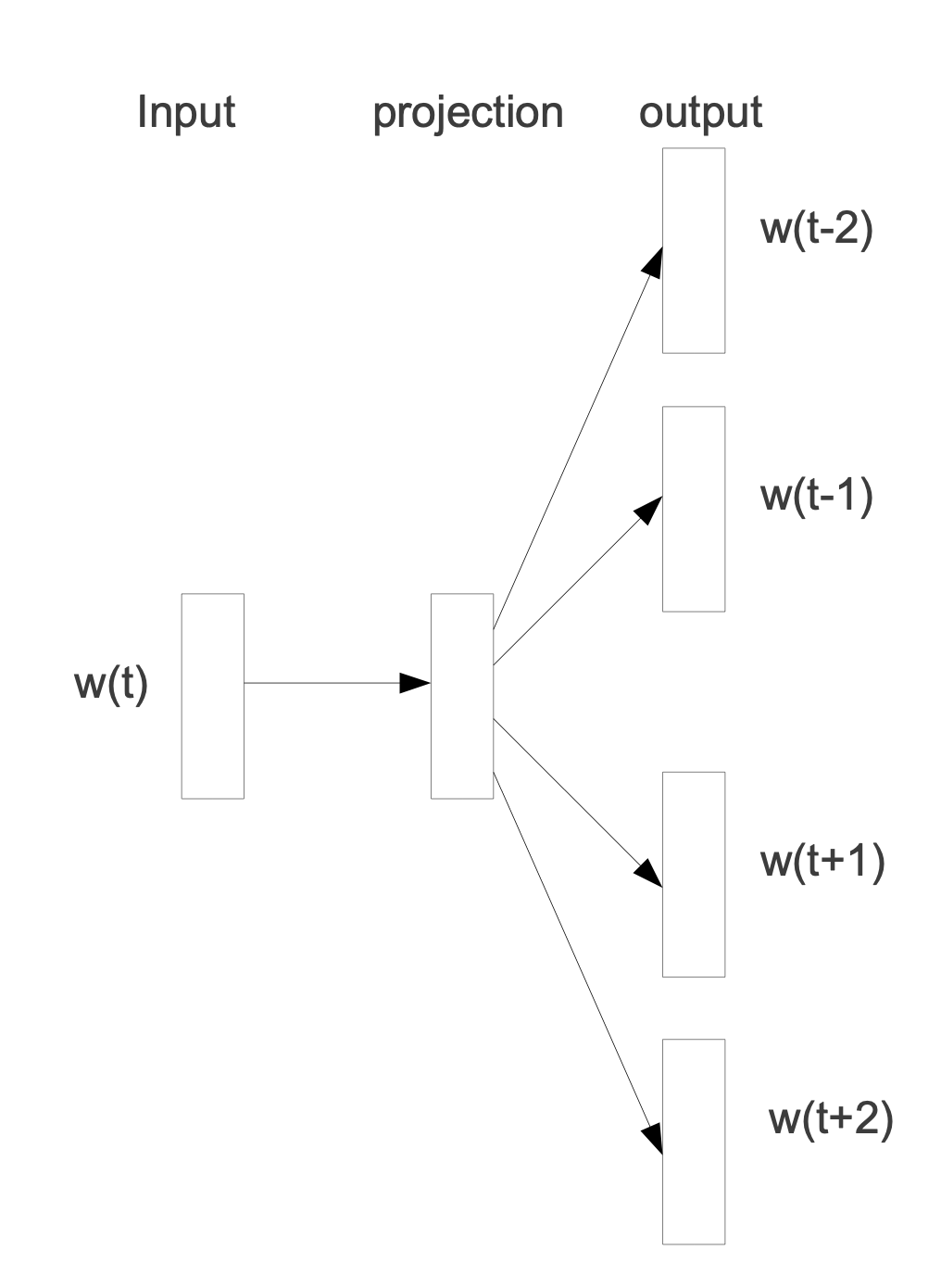


Figure 5: [Word2Vec](https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf) uses a neural network architecture. Here the input is a word in the sentence and the output are the context words that surround the input word. The projection matrix contains the vector representation of the words.

* + - [FastText](https://arxiv.org/pdf/1607.01759.pdf): This is a more advanced Word2Vec algorithm that dissects words further into letters (or combination of letters) in order to convert words into vectors. It essentially considers the internal structure of words.

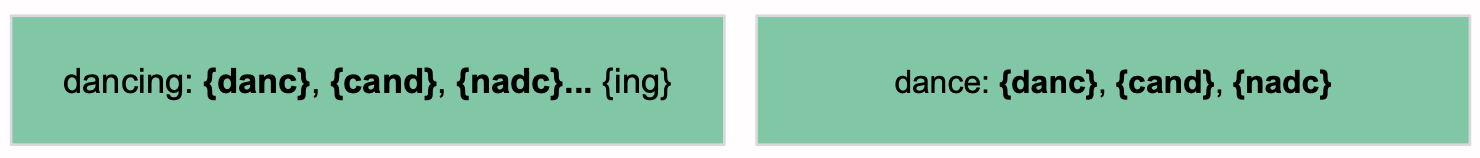


Figure 6: [FastText](https://arxiv.org/pdf/1607.01759.pdf) splits words into smaller subsets of letters. This allows it to work well on new, unforeseen words that have similar roots. In the above example if *dancing* is in the training set, and the Word2Vec model comes across *dance* it might label it as a new, unforseen word. However, FastText by splitting both into smaller subsets and also looking at the various permutations of these subsets can derive similarity between the words.

* Classification: Once we explored the various algorithms to convert the EULA texts into numbers (vectors in specific), we tested several algorithms that could correctly classify the clauses into 1) acceptable or 2) unacceptable. We have bolden the methods that we integrated in our final solution.
  + First we looked at the tradition methods:
    - **Random Forest:**
      * This algorithm trains several decision trees and averages the classification decision made by them into a single final decision about the EULA clause. A decision tree is essentially a tree-like structure with nodes representing a question (e.g. Is the clause more than 20 words?), with branches denoting the outcome of the question (e.g. Yes, No) and the root node representing the final decision made by the tree (e.g. acceptable, unacceptable). A random forest consists of several of such trees which individually try to extract different aspects of the text.

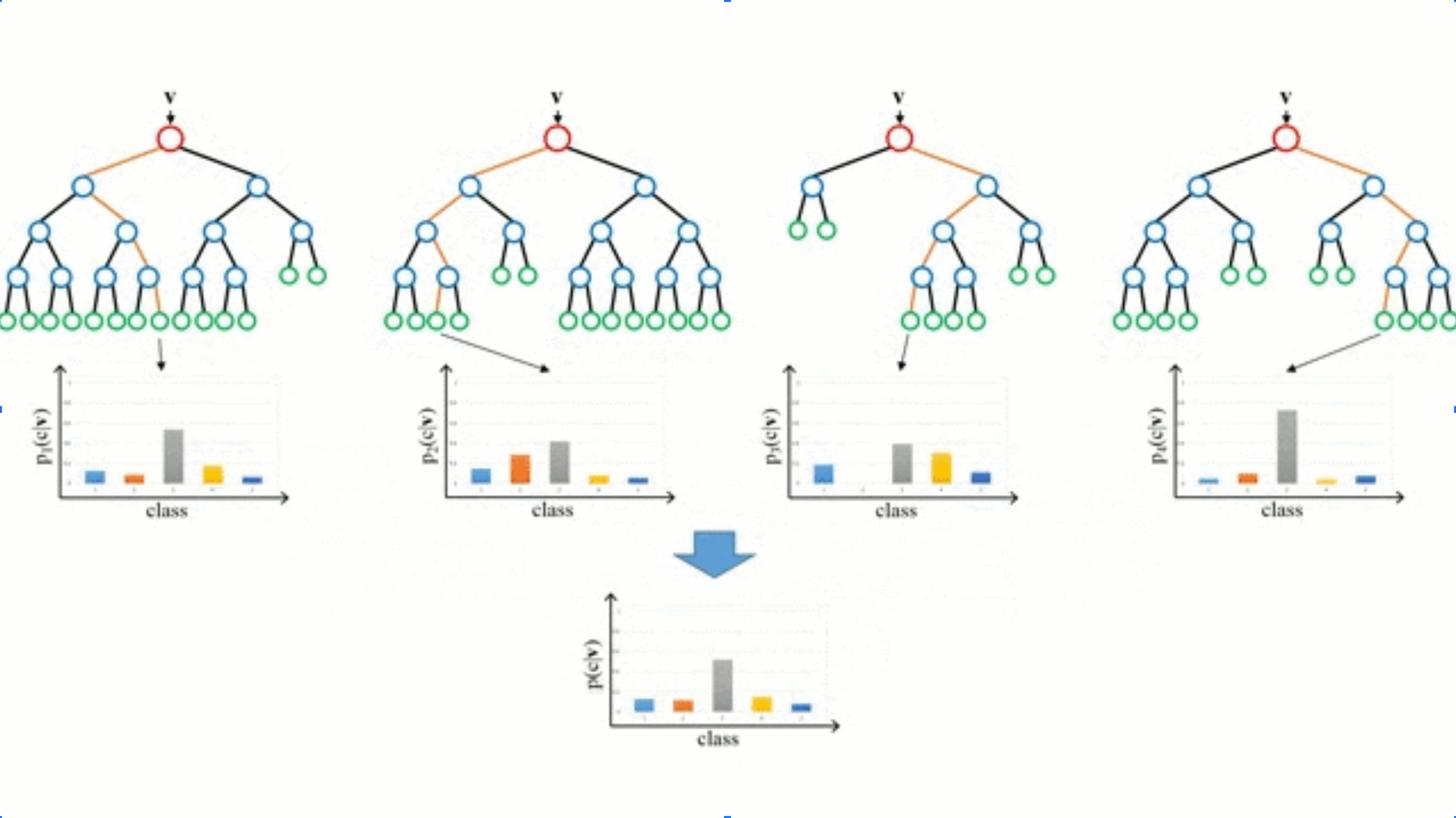
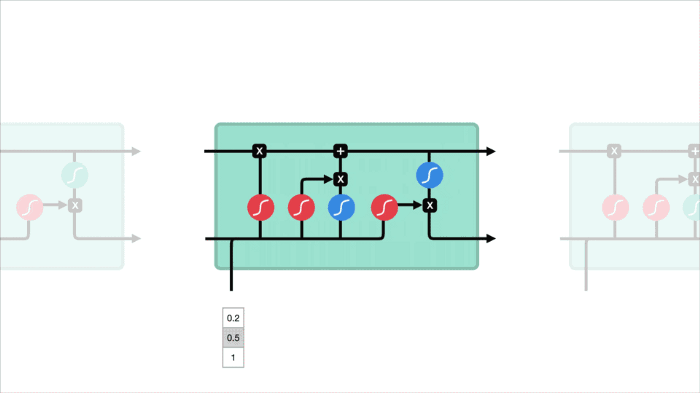


Figure 7: Illustration of 4 decision trees making individual decisions regarding the input data. The final output is the weighted sum of these decisions from each of the trees.

* + - XGBoost:
      * This is another type of tree algorithm. In this method trees are made one after the other. After each step, the data points that were misclassified by the previous tree is given a higher weightage in the next tree. In other words, each tree focuses more on the text that the previous trees did not do a good job in correctly classifying. This essentially means that, in theory, as the number of trees grows, the overall algorithm is able to classify a larger part of the dataset correctly.
  + Deep Neural Network Methods:
    - RNN-based models: LSTM, Bi-LSTM, Bi-LSTM with Attention, Bi-GRU with Attention
      * These are primarily recurrent neural networks (RNN) that capture the temporal sequence of words and infer the acceptability of the EULA text. For these types of algorithms the *sequential order of words one after the other* in a sentence is the key to extracting valuable information about the text. The first one we tried as an LSTM model.
      * Long short-term memory (LSTM): These are built on the RNN concept that capture temporal information across texts. However, the key feature is the addition of ‘memory units’ that retains information over a long period of time (in our case over a long portion of text). There are input modulation and forget gates that control this flow of information. Note: the flow of information is in the forward direction (or left to right).



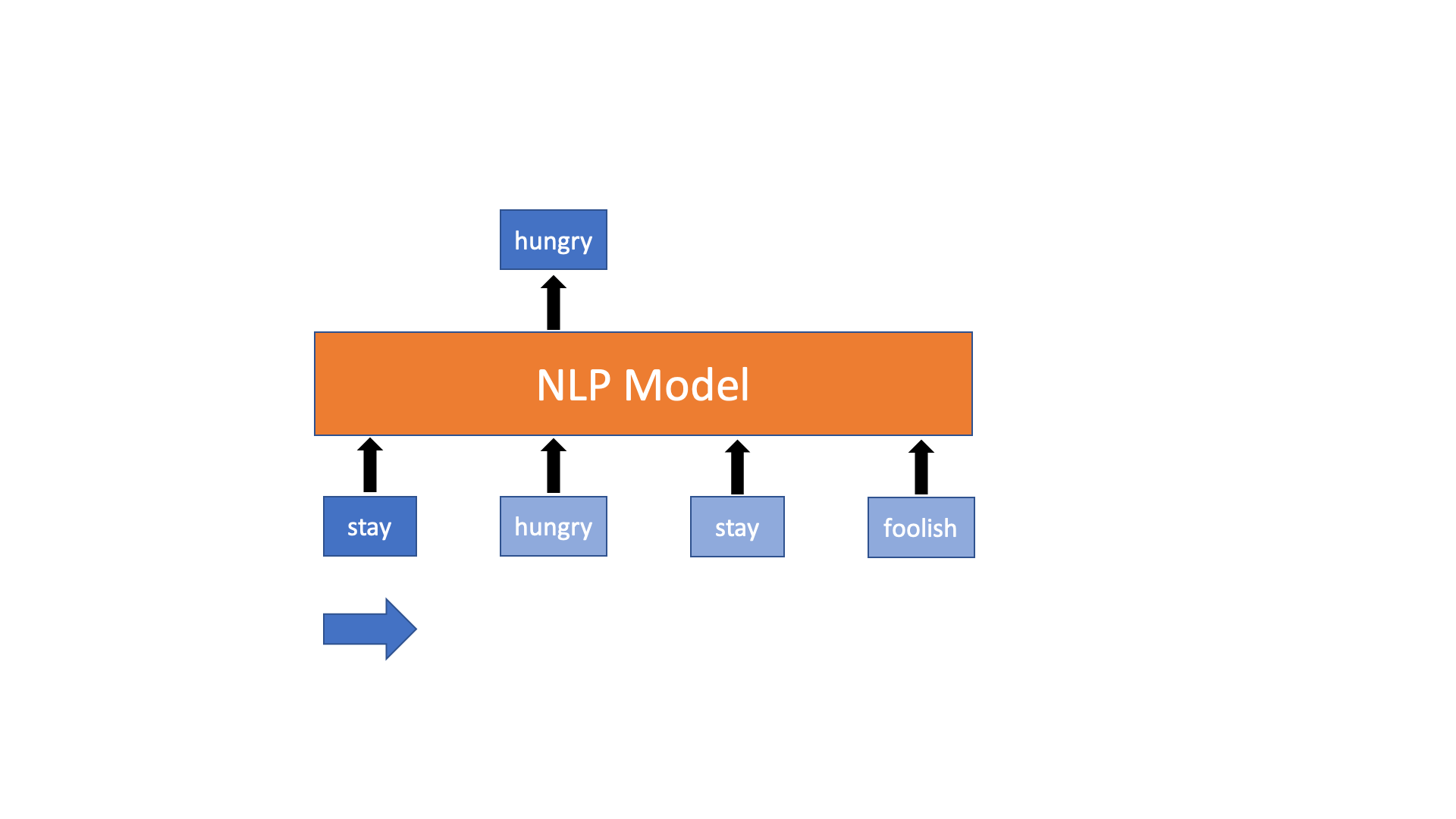


Figure 8: LSTM models go over texts in a sequential manner (from left to right). In this example we show how the model will try to predict what the next word (above NLP model) would be based on the inputs (below NLP model).

* + - * Bidirectional-LSTM (Bi-LSTM): These essentially use the same concept but here the flow of information is in two directions: 1) left to right (forward) and 2) right to left (backward). This allows the Bi-LSTM models to capture past and future information in words of the sentence to make a classification decision.

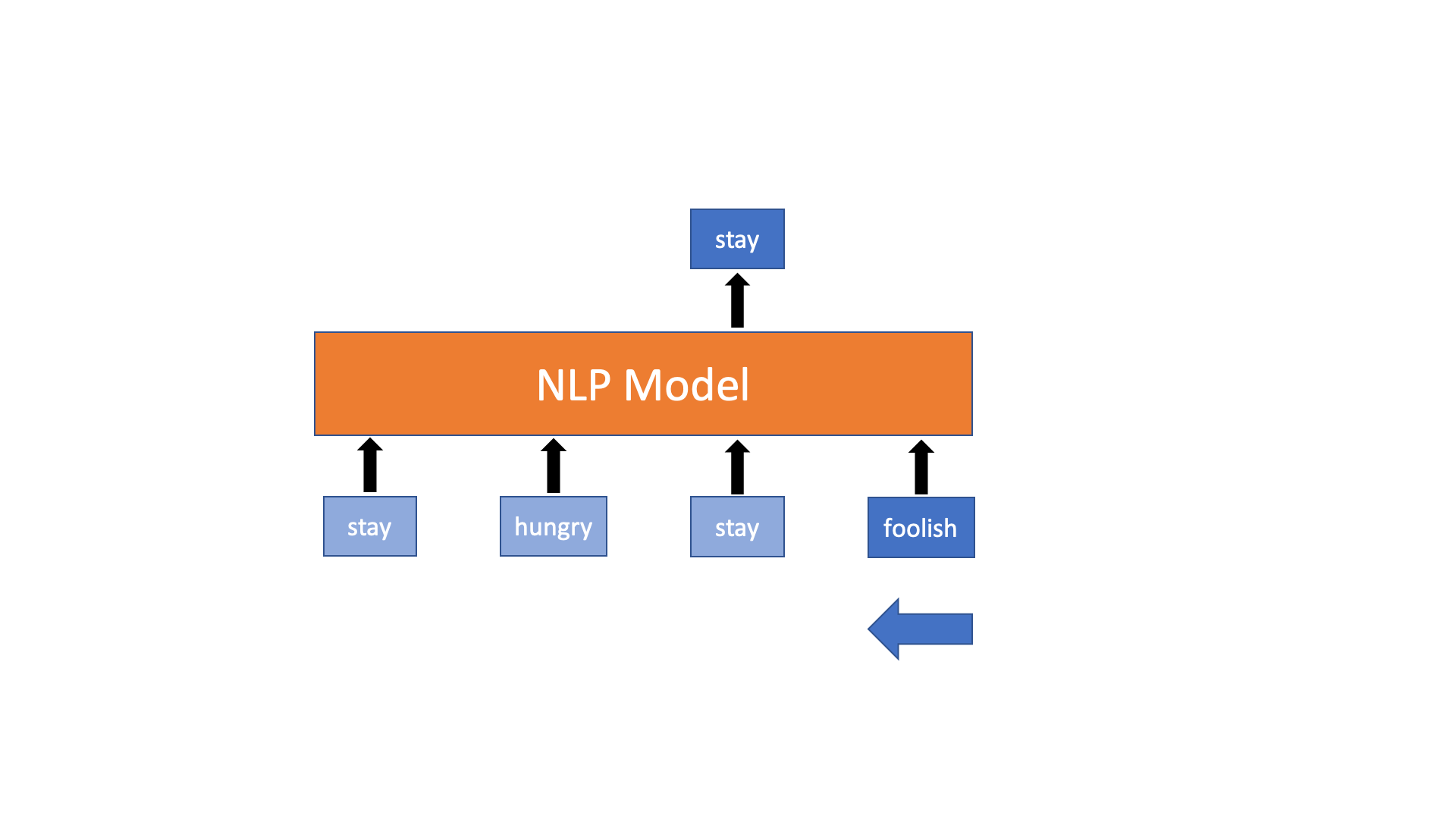
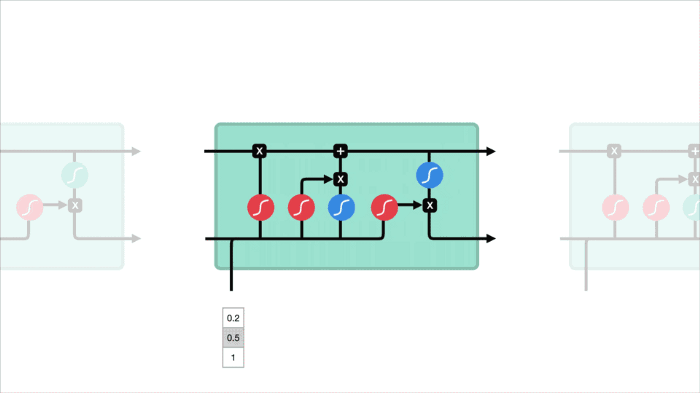
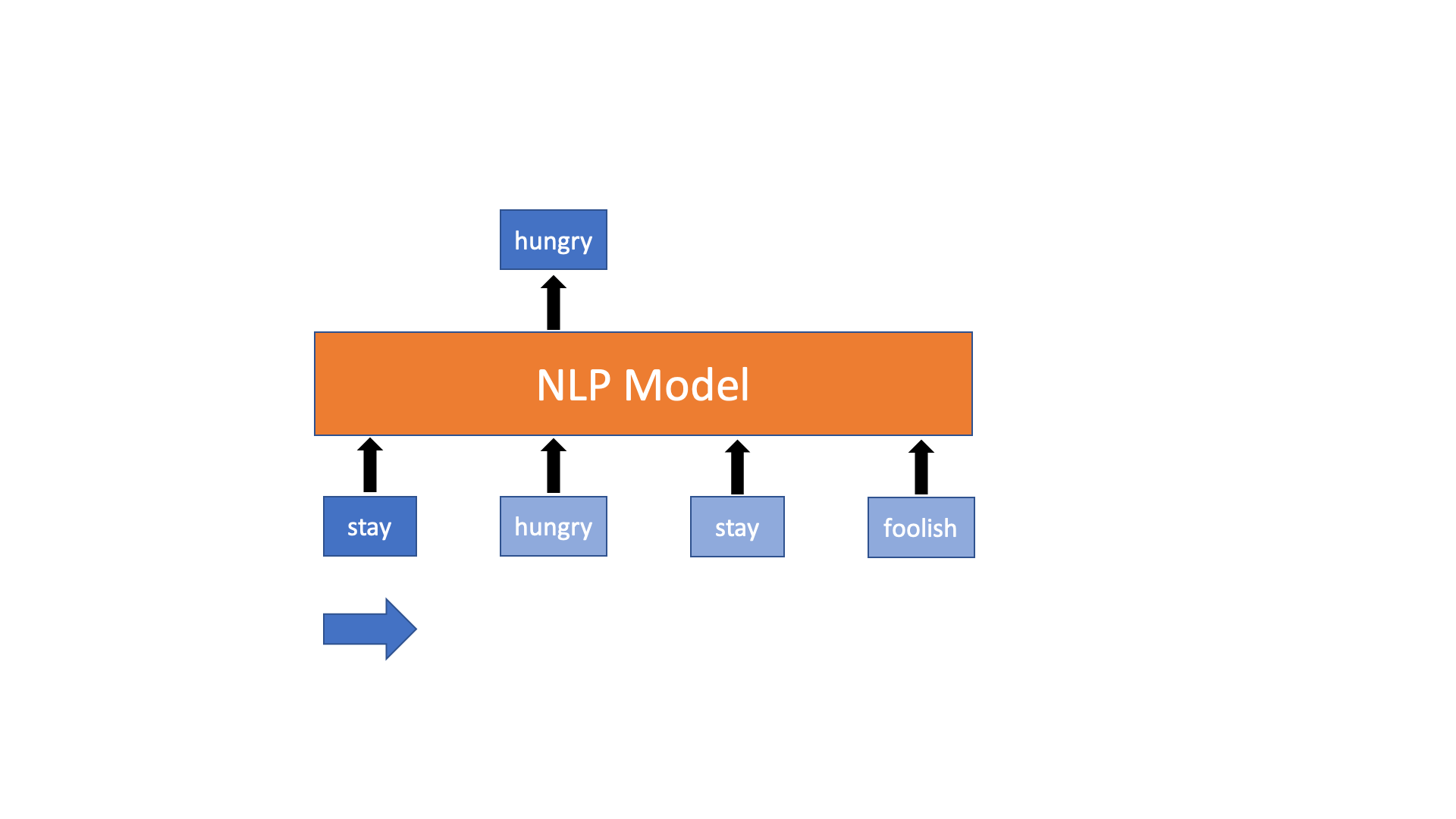
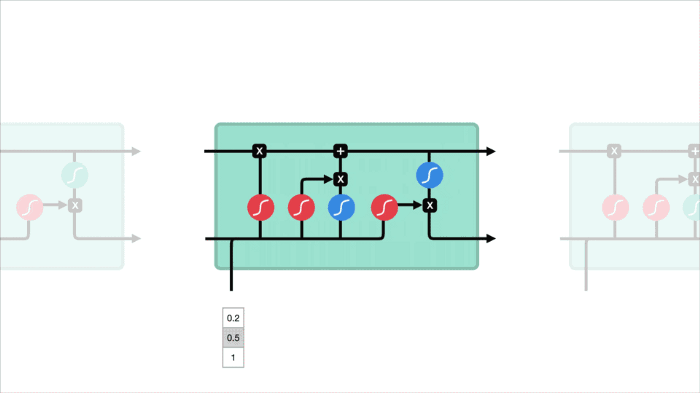


Figure 9: Bi-LSTM captures the flow of information (words in our case) from both directions.

* + - * Bi-LSTM with Attention: An attention layer allows users to know which words are most influential in making classification decisions. They also look at the influence of sequences of words with others in the text.

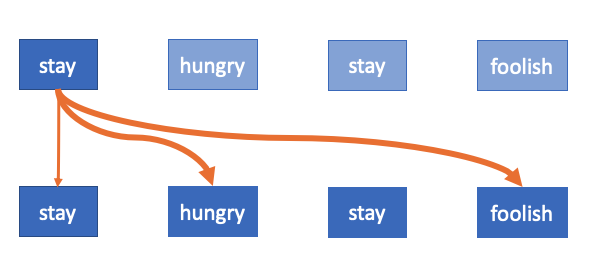
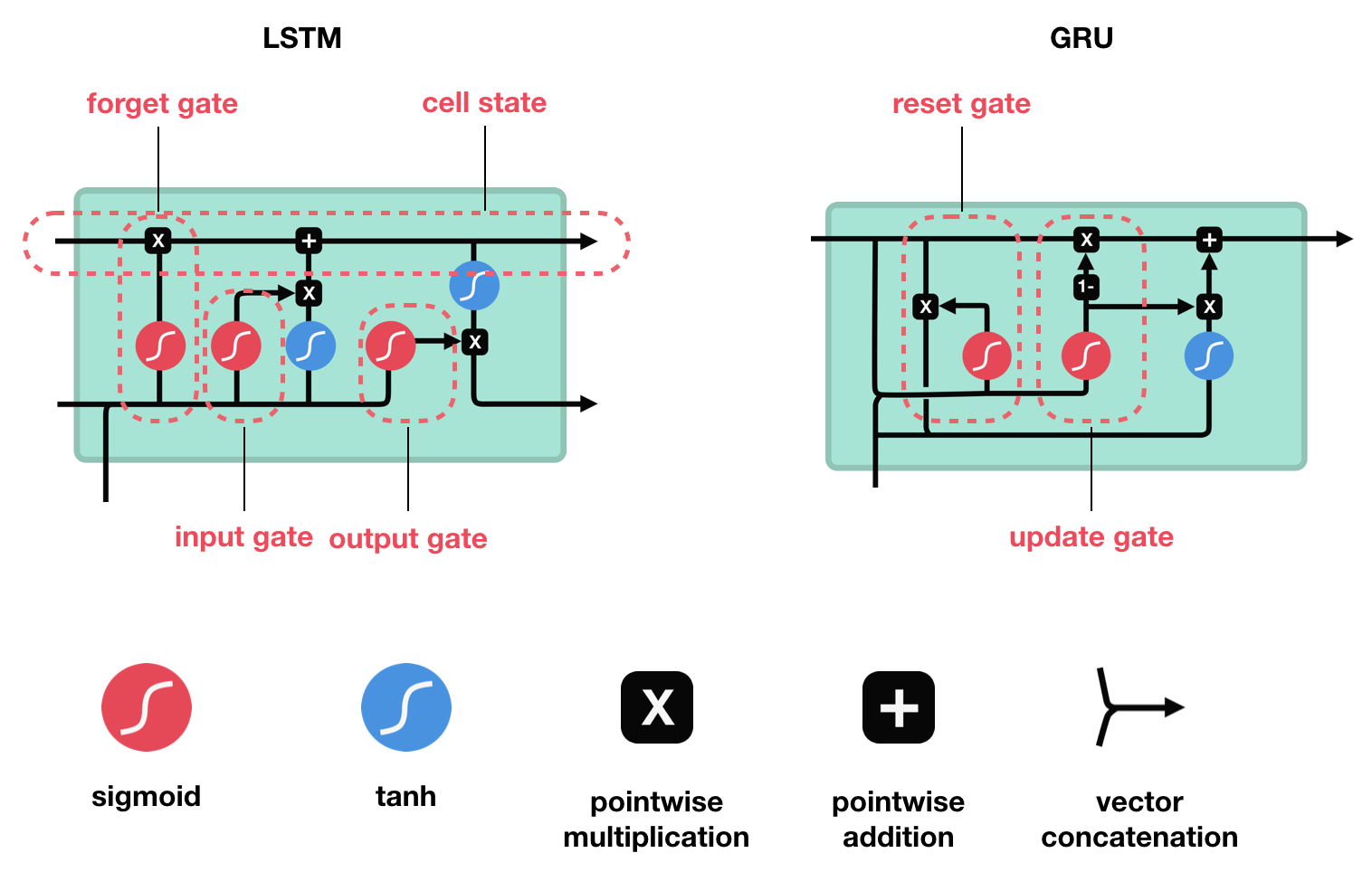


Figure 10: An attention layer provides explainability. Here the word ‘stay’ has different levels of importance to other words in the sentence. The thickness of the arrows denote these importance levels.

* + - * **Bi-Gated Recurrent Unit (GRU) with Attention**: Here we replace the Bi-LSTM nodes with GRU nodes. GRUs do not contain the forget gate but still try to capture temporal information. GRUs work well in practice in comparison to LSTMs as they have fewer parameters to train and use less memory.

Figure 11: The illustrated difference in the underlying architecture of LSTM and GRU. GRUs do not contain a forget gate and have much lower parameters to learn during the training process.

* + - [**XLNet**](https://arxiv.org/pdf/1906.08237.pdf)**:** This is the state-of-the-art algorithm in Natural Language Processing (NLP) and is essentially an auto-regressive language model that uses transformer architecture with recurrence. At a high-level, transformers are faster and robust versions of RNN that capture the temporal information in texts (sequence of words) but are faster to train. Instead of processing texts word after word (like in RNNs), these take in all texts at once and process the relationships between the words.
      * The advantage that is provided by XLNet is due to the fact that it looks that **all the possible combinations of** words in a sentence rather than simply looking at words in the forward and backward direction (like Bi-LSTMs or Bi-GRUs)

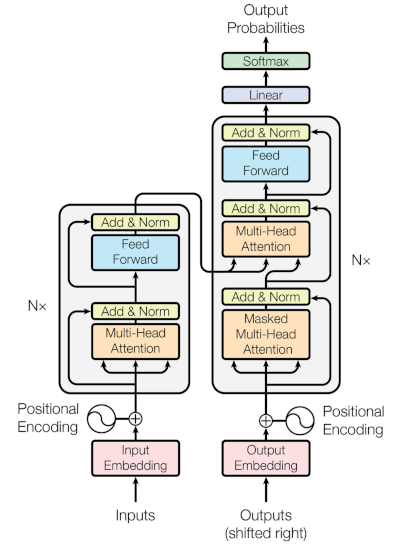


Figure 12: Example of an encoder-decoder transformer unit. This unit contains attention units within each component.

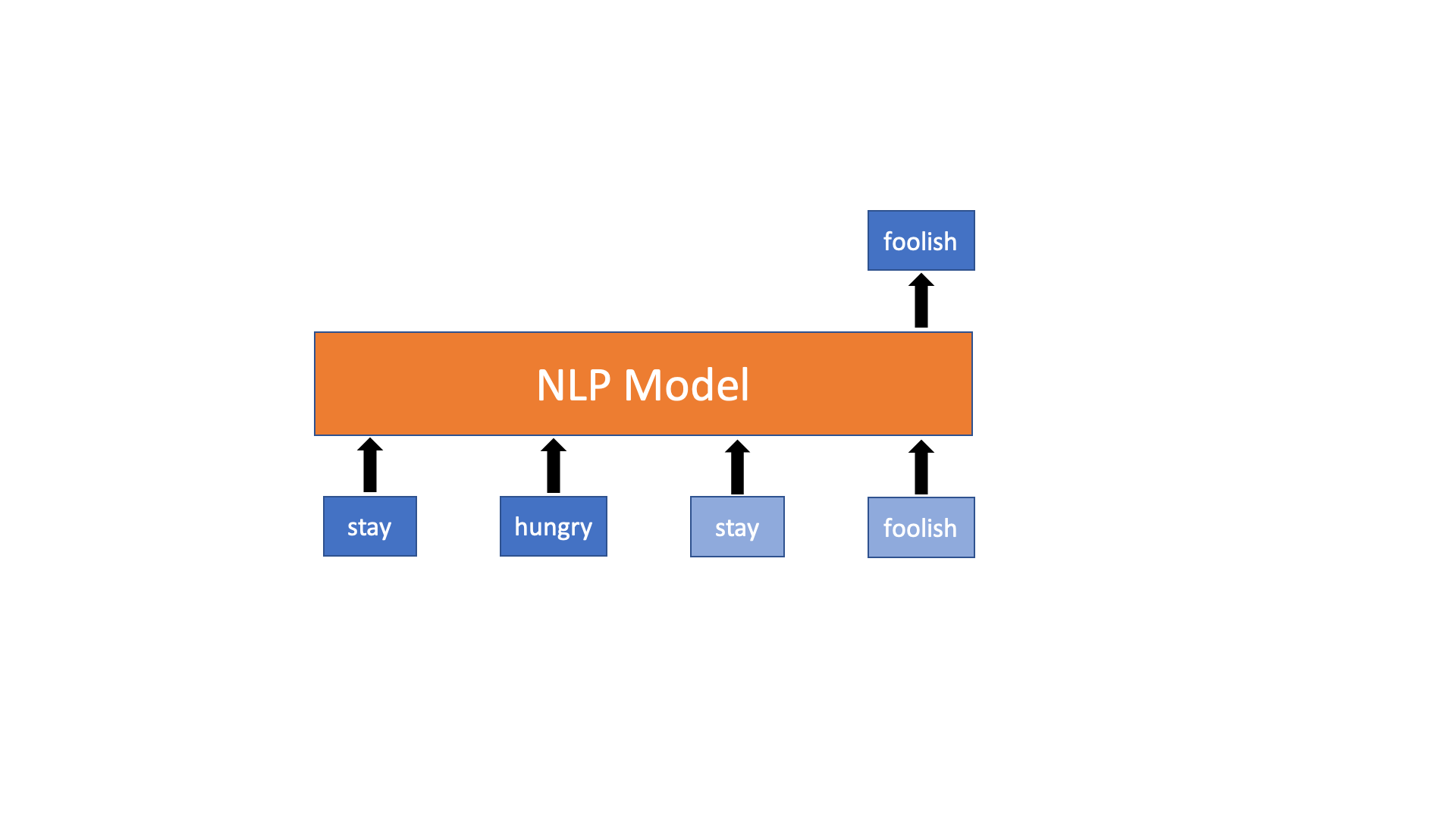


Figure 13: Several permutations of the word sequence are learned in [XLNet](https://arxiv.org/pdf/1906.08237.pdf) which allows it to capture more latent relationships between words.

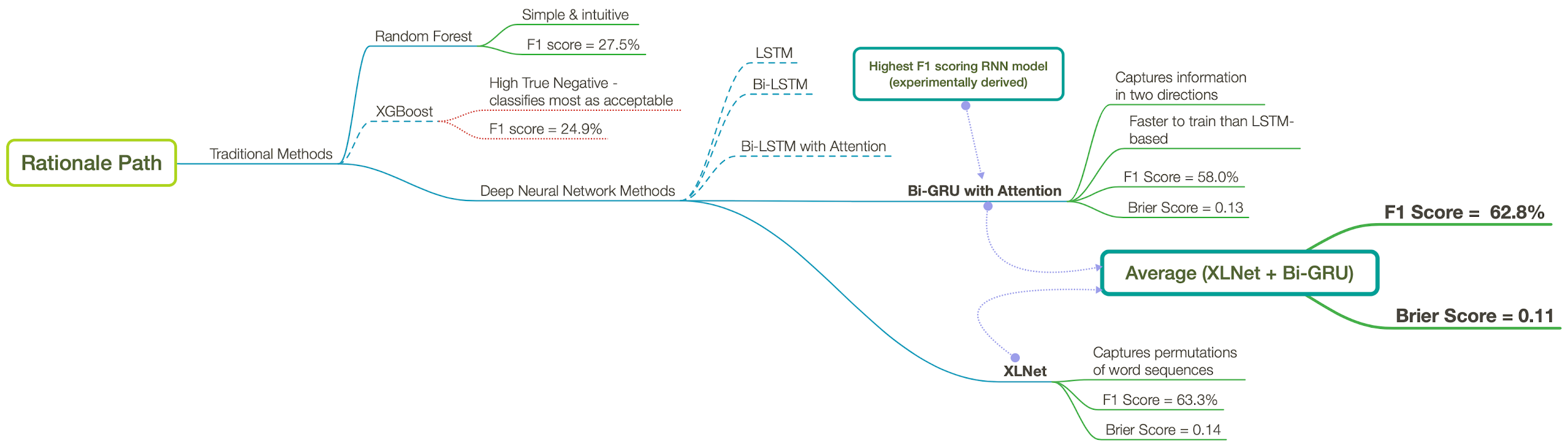
* **Transfer Learning:** We used transfer learning in the following manner:
  + **FastText**: In the process to convert text into numbers (vectors) we used FastText for our Deep Learning algorithms mentioned above. This was trained on the dataset that was provided to us for this challenge. In addition to this, we used “SRT-508 clauses” data from a prior competition to train our FastText model.
  + **XLNet**: This algorithm has been pre-trained on a massive corpus of data such as Wikipedia, BookCorpus, etc. We fed our EULA dataset to this model thereby using transfer learning to make this model more robust.

The following are the F1 and Brier Scores we achieved for the models we selected in our final solution:

**Table of F1 and Brier Score**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Random Forest (RF)** | **RNN (Bi-GRU with Attention)** | **XLNet** | **Average (XLNet + Bi-GRU)** |
| **F1 Score** | 27.5% | 58.0% | 63.3% | 62.8% |
| **Brier Score** | 0.191 | 0.131 | 0.142 | 0.116 |

**Rationale Map:**



As a first step, our team used traditional algorithms to approach the problem.

We started with **Random Forest**, which is a popular machine learning algorithm that is widely used in classification tasks.

**Random Forest** is simple and intuitive in nature. It does not require hyperparameter tuning and usually does not overfit to the dataset with an increase in the number of decision trees within the model. We achieved an F1 score of 27.5% with it.

As the next step, we implemented the **XGBoost** algorithm. XGBoost is a tree-based algorithm (like Random Forest) but uses the technique of boosting. Boosting is an error-correction algorithm which gives a higher emphasis on data-points which are misclassified. Unlike random forests, the decision trees are created iteratively, where at each step, the tree puts more emphasis on the misclassified points, so as to reduce the overall error. We therefore used XGBoost as the natural next step to Random Forest. On testing however, the F1 score achieved through XGBoost was lower than Random Forest. This was potentially due to a high False Negative, resulting in a low Recall value. This means that the model was classifying most clauses (even the ones that were labelled unacceptable) as acceptable.

After running several experiments trying to improve the accuracy of traditional models, we realized that more advanced, deep learning based models could potentially help us increase accuracies. We therefore started with the simplest form of sequence models that are used on textual data: **Recurrent Neural Networks (RNN)**. As a starting point we used a **Long Short-Term Memory (LSTM)** which attempts to resolve the vanishing gradient, a known obstacle for RNNs. LSTMs however capture the flow of information in one direction (left to right in case of sentences). **Bidirectional LSTMs** capture the flow of information from both left to right and right to left. This serves as an advantage as the model can learn from the future and the past information at a given point in the sentence. To add interpretability to our results, we added an **attention** layer that provides the importance of words in the decision making process of the model. The attention layer weighs those words differently providing more emphasis on words that have stronger relationships. Experiments run by researchers and practitioners have shown that **Gated Recurrent Units (GRUs)** train faster and provide a proportional (or sometimes better as in our case) accuracies to that of LSTMs. We therefore replaced the LSTM units with GRUs. The GRU unit does not have a forget gate (unlike the LSTM) and has fewer parameters to train on.

Arguably the most sophisticated classification model, as of now, is the **XLNet** which builds upon the transformer architecture which incorporates encoding and decoding layers in addition to sinusoidal position encoding of words in a sentence. It also incorporates permutations of words to learn more complex relationships between words. It has been pre-trained on a large corpus of data from Wikipedia, BookCorpus, etc. In our experiments, it did improve the validation accuracies.

To provide an ensemble solution, we combined our best performing models: **XLNet and Bi-GRU with attention**. The final classification decision made by this ensemble was the average of the probabilities coming out of these models. We had lively discussions about whether we should just keep XLNet or do a weighted combination of XLNet with Bi-GRUs. The result was that the argument for robustness in the estimate ensembling these two complex algorithms won the argument of the day. In addition, in our final solution, we kept the Random Forest score as a sanity check output in our display. We believe this model is the easiest to explain to lay people. However this score was not used in the average.