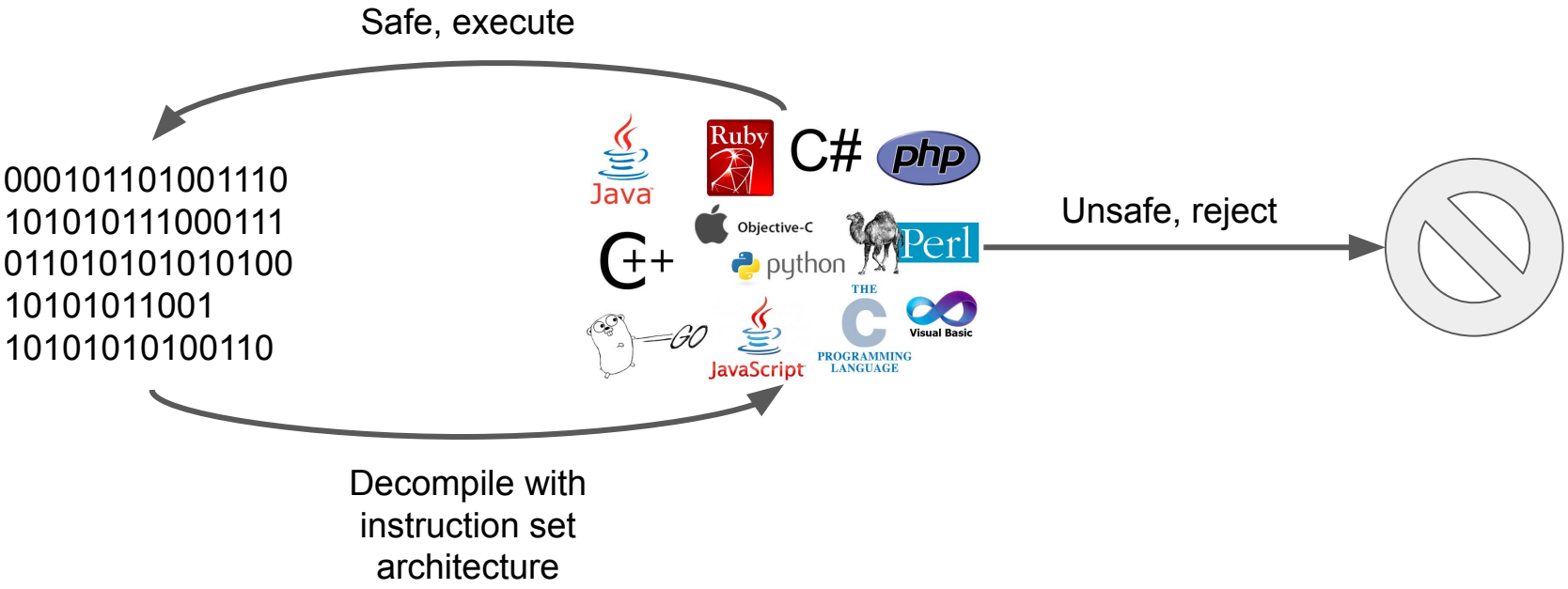
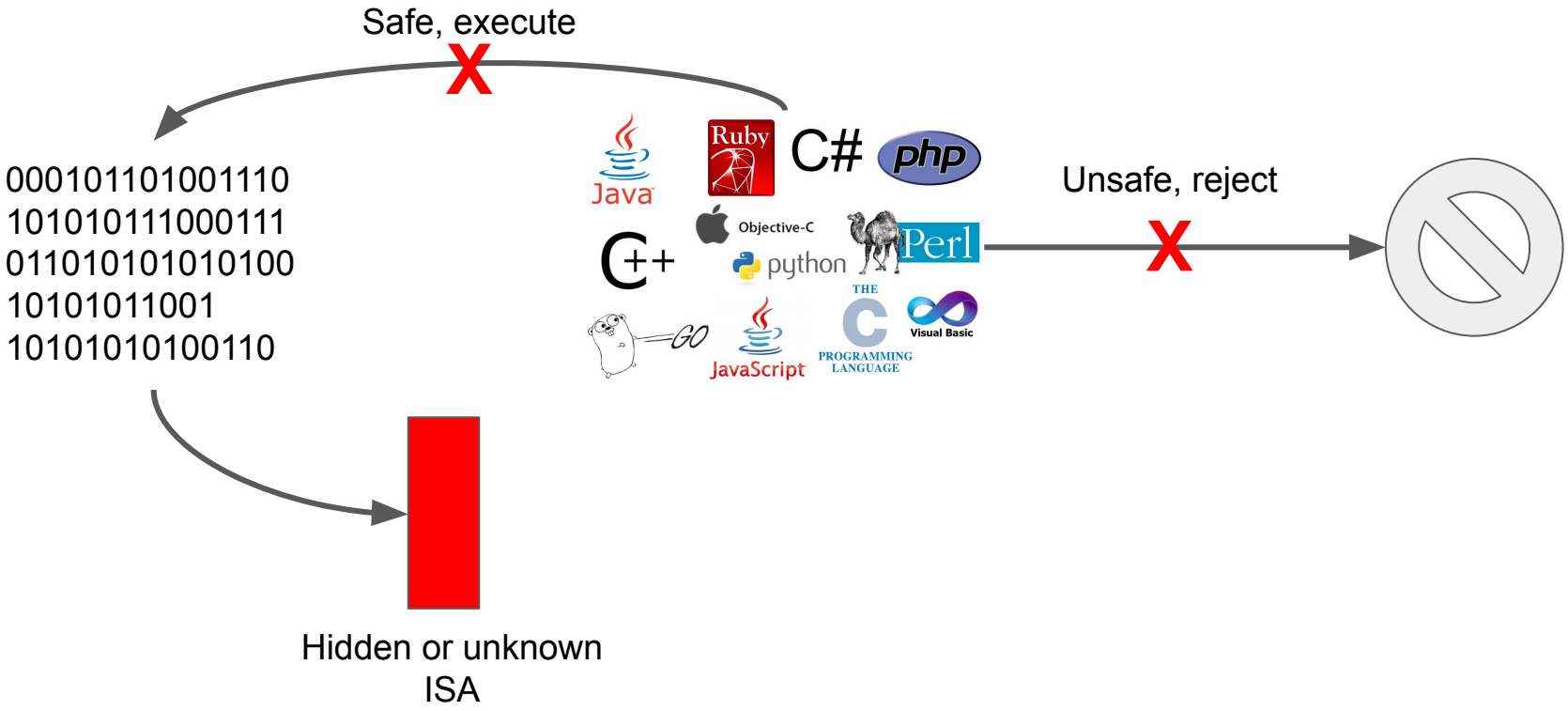
# Natural Language Processing Approaches with Ensemble Classification Models to Identify Malicious Binary Instruction

Hunter North

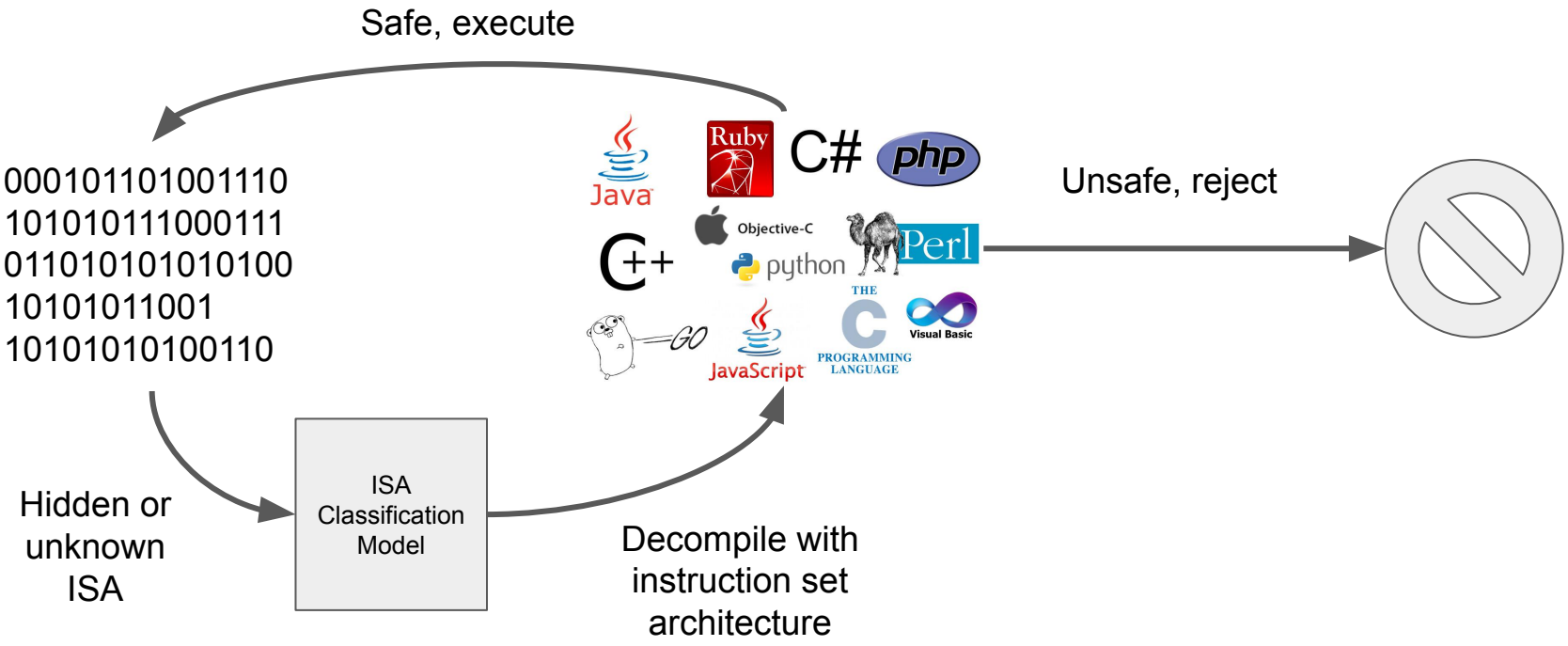
# Decompiling External Binary Files



# Decompiling External Binary Files



# Decompiling External Binary Files



# Term Frequency - Inverse Document Frequency (TF-IDF)

Term Frequency: The prevalence of a given N-gram term within a document

$$TF(\tau) = \frac{count\ of\ \tau\ in\ document}{count\ of\ all\ terms\ in\ document}$$

Inverse Document Frequency: The informativeness of a term within a collection of documents (corpus)

$$IDF(\tau) = \log\left(\frac{count\ of\ documents\ in\ corpus + 1}{count\ of\ documents\ containing\ \tau + 1}\right)$$

Term Frequency - Inverse Document Frequency: Synthesizes both term frequency and inverse document frequency into a single feature that marks the prevalence of a given N-gram but will scale the overall feature by the N-grams informativeness

$$TF - IDF(\tau) = TF(\tau) \cdot IDF(\tau)$$

Use of TF-IDF to classify instruction set architecture was developed by Dinuka Sahabandu in “A Natural Language Processing Approach for Instruction Set Architecture Identification”

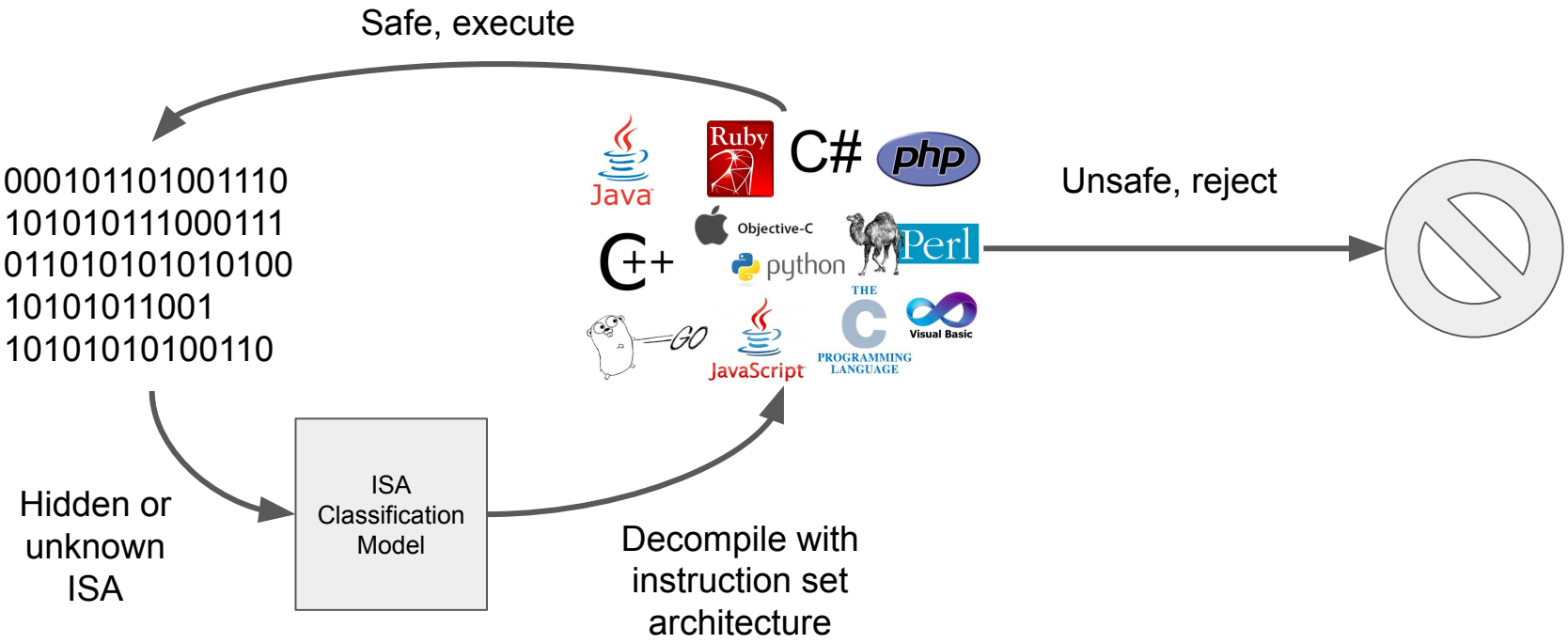
# TF-IDF Dimensional Scoring

I have slightly adjusted Dinuka's work here to capture a bit more granularity. There is no longer encoding of the binary file which allows us to pick up more unique lengthed features.

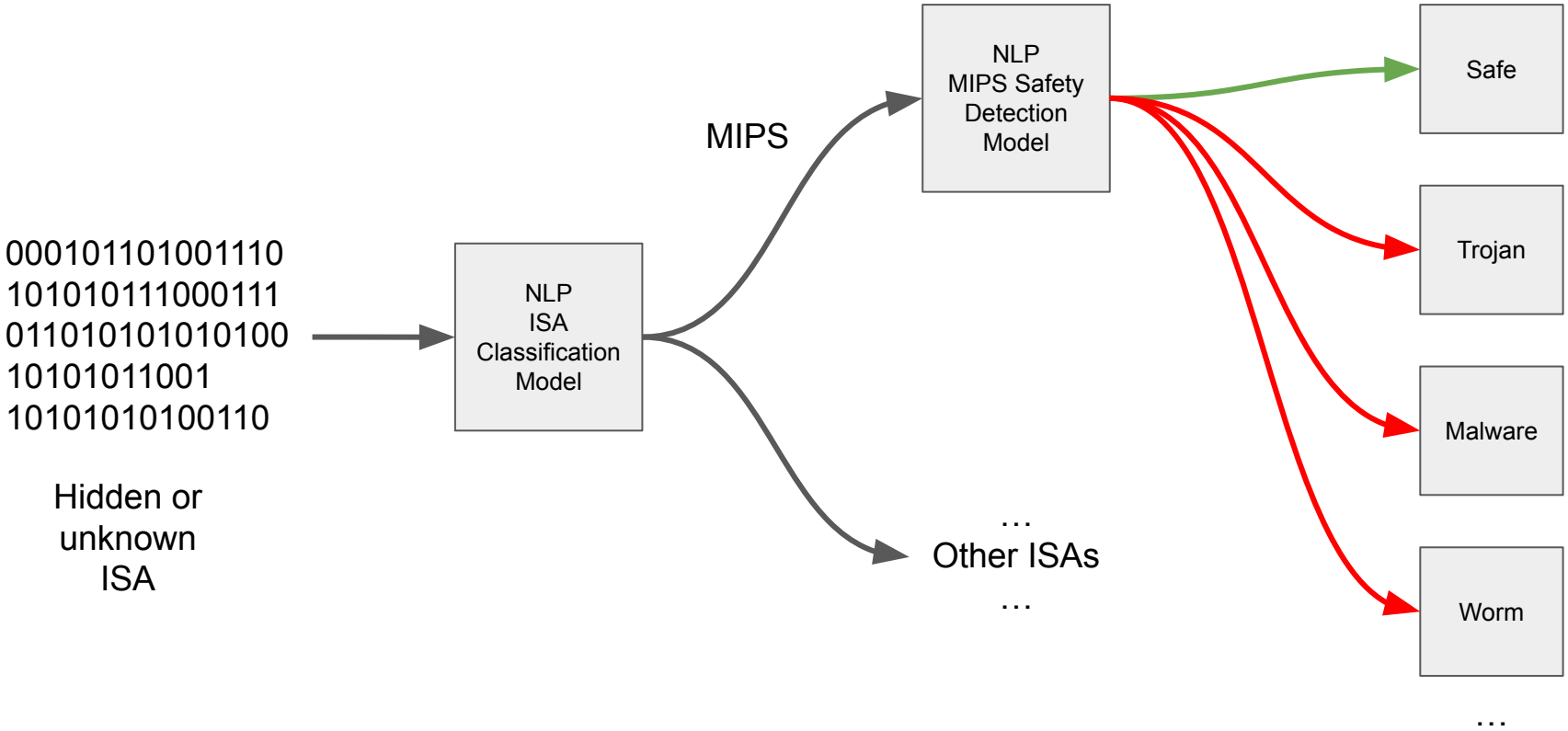
Accuracy of models built on training models with all the possible TF-IDF features corresponding to specific N-gram dimensions:


Use of TF-IDF to classify instruction set architecture was developed by Dinuka Sahabandu in "A Natural Language Processing Approach for Instruction Set Architecture Identification"

# Initial Problem



# Skipping the Decompilation Step (Harder)



# Periodic Tracing (PT) Version One

Previous natural language processing approaches (TF-IDF) to classify instruction set architecture fail to capture the characteristics of how an N-gram term is being used **within** a specific document.

We can fulfill this by collecting a vector holding the spacing (or count of terms) between each consecutive use of an N-Gram. Naming this vector V, we can create two new features:

Periodic Length: The average amount of spacing that is occurring between occurrences of consecutive N-gram terms

$$PL(\tau) = \overline{V_\tau}$$

Periodic Variance: The variance in the spacing of consecutive N-gram terms

$$PV(\tau) = \sigma\left(V_\tau\right)$$

# Periodic Tracing (PT) Version One

Here I seek to merge the two previous types of periodic tracing features into a singular one. As mentioned earlier, we begin by forming a vector ($PT_{\tau}$) of all the spacings between consecutive N-Gram terms ($\boldsymbol{\tau}$).

Periodic Length: The average amount of spacing that is occurring between occurrences of consecutive N-gram terms

$$PL(\tau) = \overline{V}_{\tau}$$

Trust Scalar: The amount of trust we can assign to the measured periodic length. Will approach zero as the variance of word spacing grows large.

$$TS(\tau) = \frac{1}{1 + \sigma\left(PT_{\tau}\right)}$$

Periodic Tracing: The spacing and consistency of specific of an N-gram term throughout a document.

$$PT(\tau) = PL(\tau) \cdot TS(\tau)$$

# Periodic Tracing (PT) Version Two

In this iteration of the periodic tracing feature category that will allow a tunable sensitivity property to the trust scalar. Both the periodic length and periodic tracing equations stay the same, but the trust scalar is now:

$$TS(\tau) = \frac{1}{1 + \sigma\left(PT_{\tau}\right)^{w}}$$

Here is how the sensitivity tuning affects the periodic tracing feature (with a periodic length of 10):

# Main Idea Behind Tuning Parameter

$$PT_1(mean,\ var) = mean \cdot \frac{1}{1+var}$$

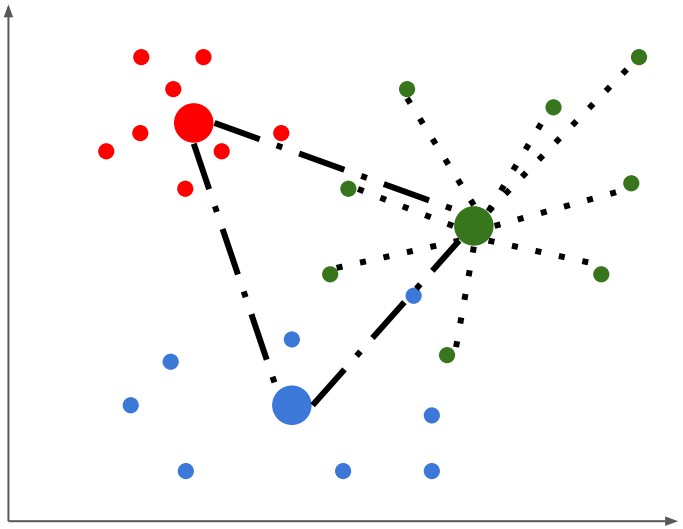
$$PT_1(8,\ 3) = 8 \cdot \frac{1}{1+3} = 2 \qquad PT_1(16,\ 7) = 16 \cdot \frac{1}{1+7} = 2 \qquad PT_1(8,3) = PT_1(16,\ 7)$$

$$PT_2(mean,\ var) = mean \cdot \frac{1}{1+var^w}\ ;\ say\ w = 1.3$$

$$PT_2(8,\ 3) = 8 \cdot \frac{1}{1+3^{1.3}} = 1.918 \qquad PT_2(16,\ 7) = 16 \cdot \frac{1}{1+7^{1.3}} = 1.27 \qquad PT_2(8,3) \neq PT_2(16,\ 7)$$

Periodic Tracing V3 (with a proper omega value) means less collisions and more spaced classes!

# Calculating the Tuning Parameter



$$class - cluster(\ c\ ) = \sum_{i=0}^{size(\ c\ )} \left| c_i - \bar{c} \right|$$

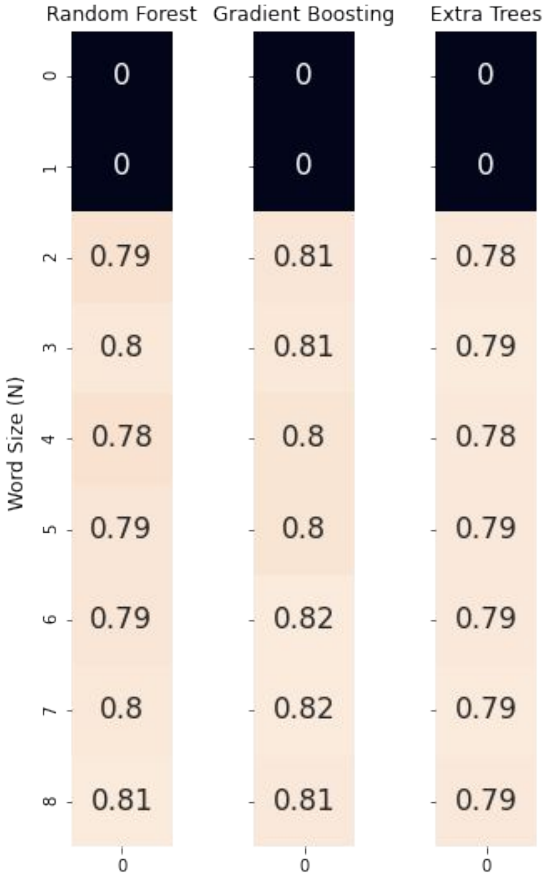
$$set - cluster(\ \ ) = \sum_{i=0}^{size(\ c\ )} \sum_{j=i+1}^{size(\ c\ )} \left| \bar{j} - \bar{c} \right|$$

$$\omega \Rightarrow maximize\left( \frac{set - cluster(\ \ )}{\sum_{c\ =\ 0}^{size(\ c\ )} class - cluster(\ c\ )} \right)$$

Could maybe use two different tuning parameters, one for each type of cluster

# PT (V2) Dimensional Scoring

Accuracy of models built on training models with all the possible PT (version 2) features corresponding to specific N-gram dimensions:



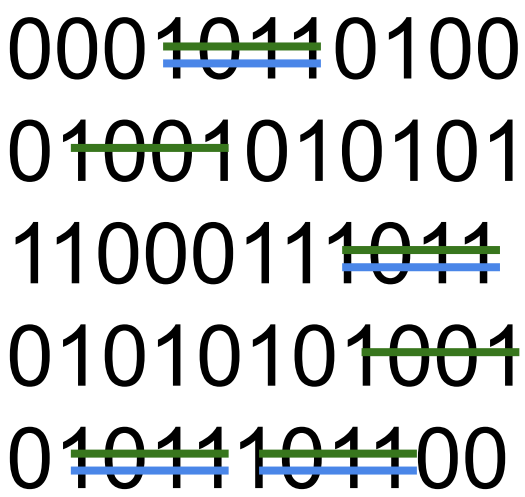

Have not applied the TF-IDF filter into ISA class before training/testing periodic tracing

Previous NLP approach (TF-IDF) scores around 60 percent

# Theoretical ideas that could compensate for periodic tracing shortcomings

# Modifying N-Gram Feature Space With M-Skips

With a better understanding of computer architecture, one will recognize why it might be favorable to reconfigure n-gram terms to include skip terms located within each term. An example of how this would work:




M = 0, N = 4 term: 1011

M = 1, N = 4 term: 10_1

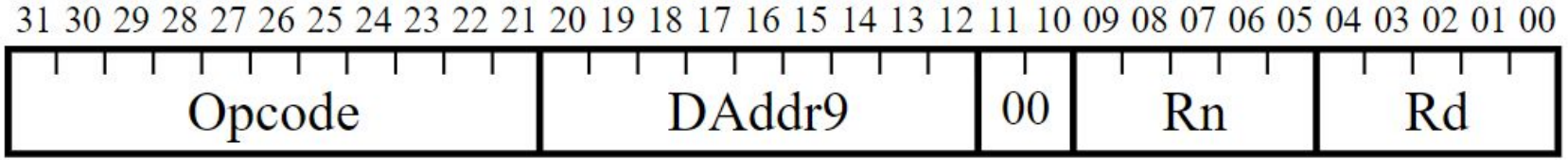
Captures 1001 and 1011

Why might this be favorable…

# Modifying N-Gram Feature Space With M-Skips

In assembly instruction, separate parts of an instruction will have correlation. For example, this D-Type instruction below (ARM ISA) will always have a ‘00’ existing 9 bits after the opcode. This characteristic of the instruction set can be leveraged to avoid potential conflation between the existence of the opcode term being used as an ARM D-Type opcode with the opcode term being used for some other reason (like an address, immediate value,  or an opcode in another instruction set).

If we insert 9 skips into our term, we can now trace the term “{Opcode}{9 skips}00”



# Reduction of Large Feature Space

Without Skip Terms:

$$TF-IDF_{feature\ space}(1\ to\ n)=\sum_{i=1}^{n}(2^{i})$$

$$PT_{feature\ space}(1\ to\ n)=\sum_{i=1}^{n}(2^{i})$$

Applying Skip Terms:

$$TF-IDF_{feature\ space}(1\ to\ n)=\sum_{i=1}^{n}\left(4\cdot 3^{i-2}\right)$$

$$PT_{feature\ space}(1\ to\ n)=\sum_{i=1}^{n}\left(4\cdot 3^{i-2}\right)$$

Too many features for building and testing models…

# Large Feature Space Reduction: Modified Boruta Approach

| binary | ISA | feature | feature_2 | feature_3 | feature_4 | feature_5 |
|---|---|---|---|---|---|---|
| $x_1$ | $y_1$ | $z_1$ | $z_2$ | $z_1$ | $z_3$ | $z_5$ |
| $x_2$ | $y_2$ | $z_2$ | $z_4$ | $z_2$ | $z_5$ | $z_1$ |
| $x_3$ | $y_3$ | $z_3$ | $z_1$ | $z_3$ | $z_1$ | $z_2$ |
| $x_4$ | $y_4$ | $z_4$ | $z_5$ | $z_4$ | $z_2$ | $z_4$ |
| $x_5$ | $y_5$ | $z_5$ | $z_3$ | $z_5$ | $z_4$ | $z_3$ |

Create 4 random permutations of our feature in question → Train and score 5 models in predicting ISA from each of the feature columns → The feature is important if it's score beats all 4 of the models based of random permutations → Assign a score to the important feature:

$$acc - \overline{random\ acc}$$