Machine Learning for INSPIRE's Core Predictor

Andrew Blance,
Aidan Sedgewick &
Parisa Gregg

Durham University

Outline

- Explain the ‘Core predictor’ problem
- Describe the data
- Methods used
  - Machine learning schemes
  - Text-based features
  - Article reference features
- Constructing a solution (SVM classifier based on references)
- Confidence in solution
- Conclusions

Problem/Task

- INSPIRE is the High-Energy Physics Literature Database
- Content is taken from many sources, including arXiv
- Database is updated daily, with articles classified as “Core” or “Non-Core”, or are rejected for inclusion.
- Is there a way to automate this?

Non-Core example

The Moutard Transformation of Two-Dimensional Dirac Operators and Möbius Geometry

I. A. Taimanov*

Sobolev Institute of Mathematics, Russian Academy of Sciences, Novosibirsk, Russia
Novosibirsk State University, Novosibirsk, Russia

Received August 6, 2014

Abstract—We describe the action of inversion on given Weierstrass representations for surfaces and show that the Moutard transformation of two-dimensional Dirac operators maps the potential (the Weierstrass representation) of a surface S to the potential of a surface $\widetilde{S}$ obtained from S by inversion.

Core example

R-symmetric high scale supersymmetry

James Unwin*

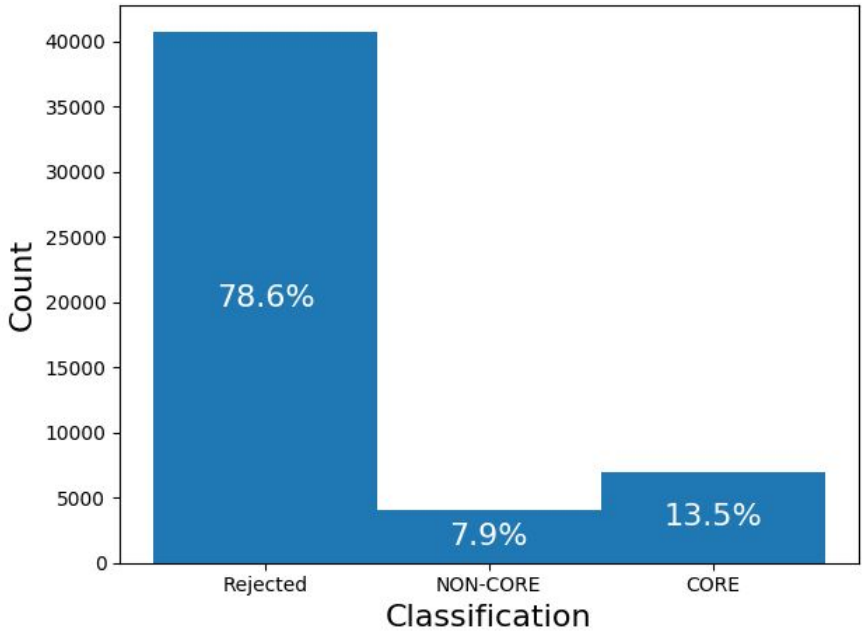
Mathematical Institute, University of Oxford, 24-29 St. Giles', Oxford OX1 3LB, United Kingdom and Rudolf Peierls Centre for Theoretical Physics, University of Oxford, 1 Keble Road, Oxford OX1 3NP, United Kingdom

(Received 10 October 2011; revised manuscript received 11 September 2012; published 1 November 2012)

Introducing an R-symmetry to models of high scale supersymmetry (SUSY) can have interesting consequences, and we focus on two aspects. If Majorana masses are forbidden by an R-symmetry and the main source of electroweak gaugino masses are Dirac terms, then the Higgs quartic coupling vanishing at the SUSY scale and the Higgs boson mass will be near 125 GeV. Moreover, using an R-symmetry, models with only one Higgs doublet in the UV can be constructed and we argue that, since we desire only a single Higgs at the weak scale, this scenario is more aesthetic than existing models. We subsequently present a model which draws on both of these features. We comment on neutrino masses and dark matter in these scenarios and discuss how the models presented can be discerned from alternative constructions with high scale SUSY, including split SUSY.

Gathering Data

- Gathered all arXiv listings (including updated articles) 1/1/16 - 31/5/16.
- 52,000 articles, strongly skewed.
- Shuffle, and divide the data for Training:Validation:Testing.
- DO NOT LOOK AT THE TESTING DATA!



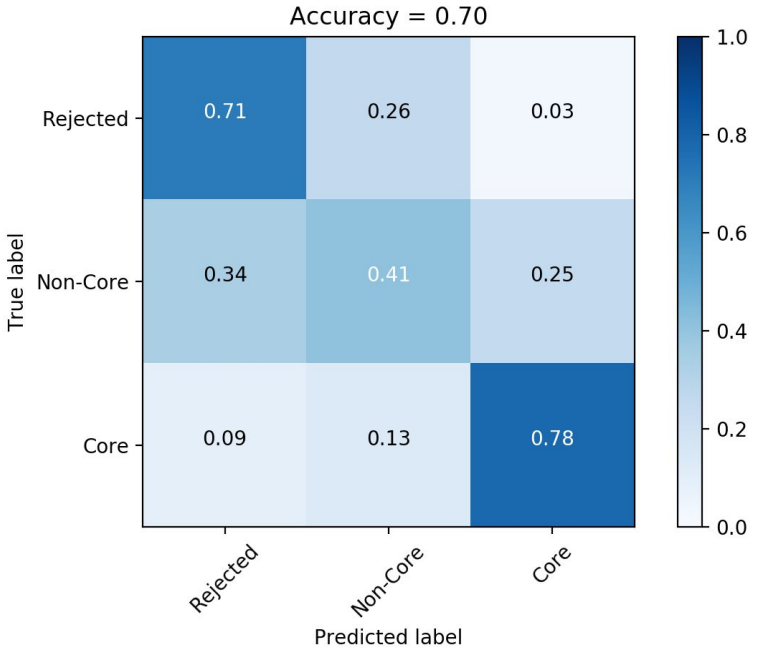
Machine Learning

- We used sklearn[1] and keras[2]
- Many different algorithms to choose from eg. svm, knn, naive bayes
- You train the algorithm on different “features” of the data
- Features can be stuff like the words in the text, the authors of the paper, the references of the paper.

[1] - https://arxiv.org/pdf/1201.0490.pdf
[2] - https://github.com/keras-team/keras

INSPIRE's initial algorithm

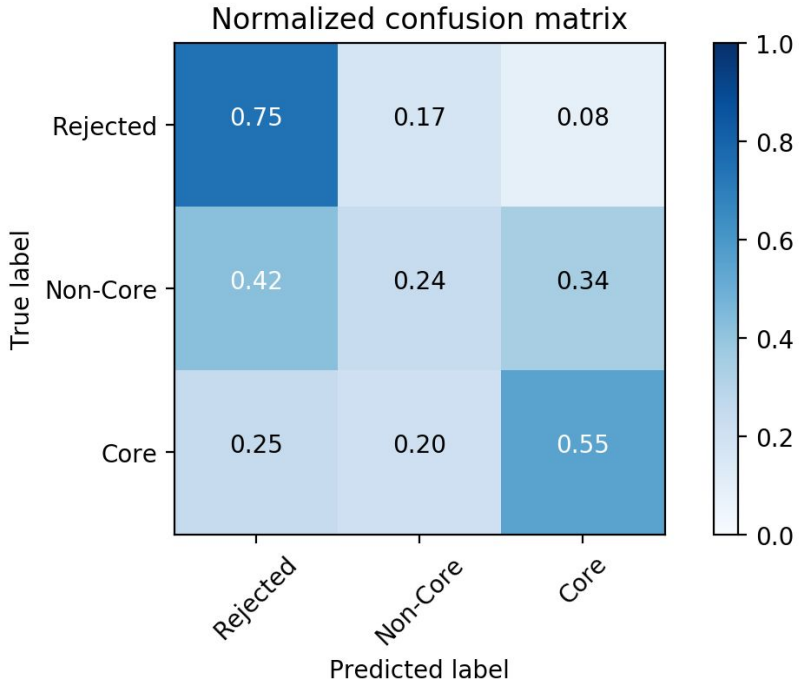
- This shows the performance of the initial algorithm Inspire have developed
- This gives an idea of the baseline which we want to improve upon



Dictionary of keywords

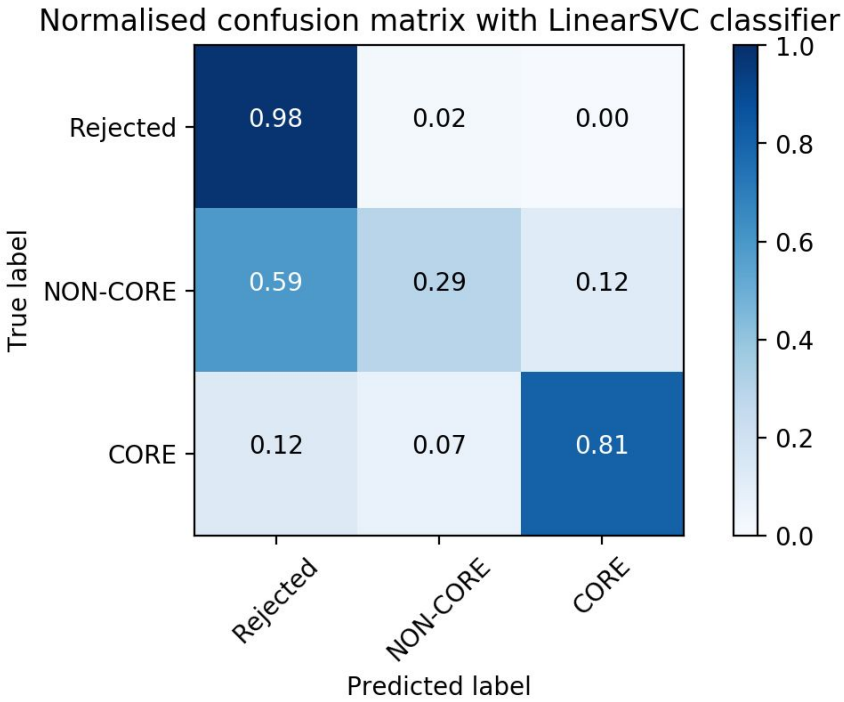
- Used dictionary of HEP words and terms
  - Unigrams: higgs
  - Bigrams: charged current
  - Trigrams: muon tracking detector
  - Quadgrams: inclusive reaction central region
- Counted frequency of these words in title and abstract
- Trained SVM on these features

Accuracy: 0.70



Bag of words

1. Turn string of text into list of ‘tokens’
2. Count frequency of each ‘token’
3. Normalise with text length and overall frequency in the corpus - TFIDF
4. Token frequency is treated as a feature
5. Each text corresponds to a vector of word frequencies
6. Classifier is then trained on a matrix of n_tokens x n_texts

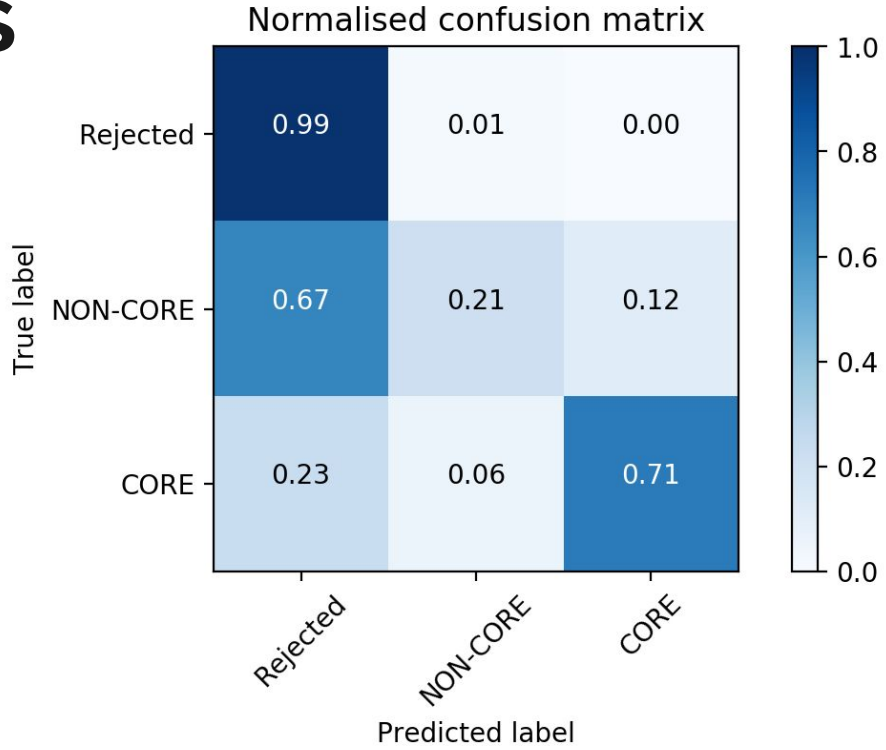


Accuracy: 0.92

Word embeddings

- Here, words are represented as vectors
- Words with similar context will have vectors ‘close’ to each other
- How did we get the mapping?
  - Embeddings layer in a NN using keras
  - Using pre-learned GloVe embeddings
- However, both these techniques gave similar results

Accuracy: 0.93



Performance trends so far

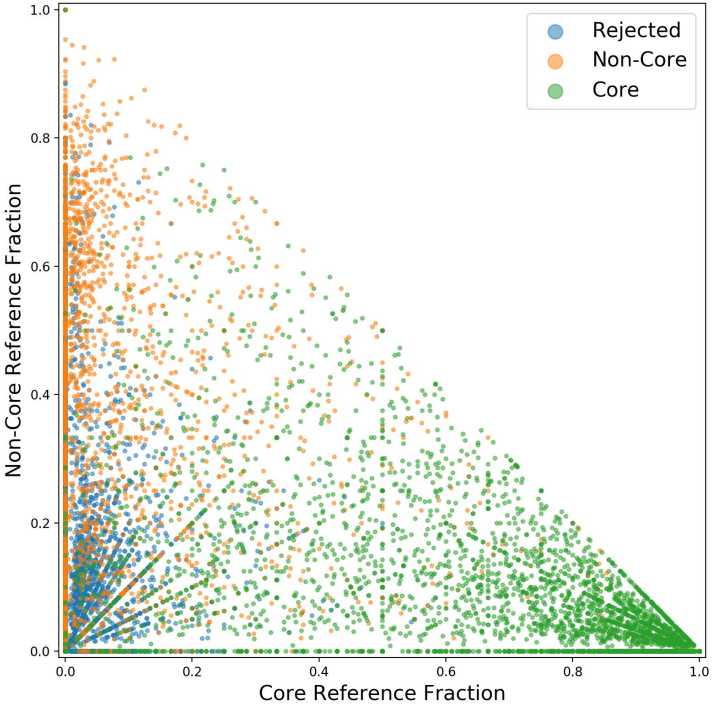
- Each method has improved upon the last:
  - Dictionary < bag of words
  - Bag of words < word embeddings
- However, the common limiting issue is the classification of Non-Core
- Time to try a new feature!

Reference fractions

- If there are *N* references in a paper, and *A* are Core papers, *B* are Non-Core,

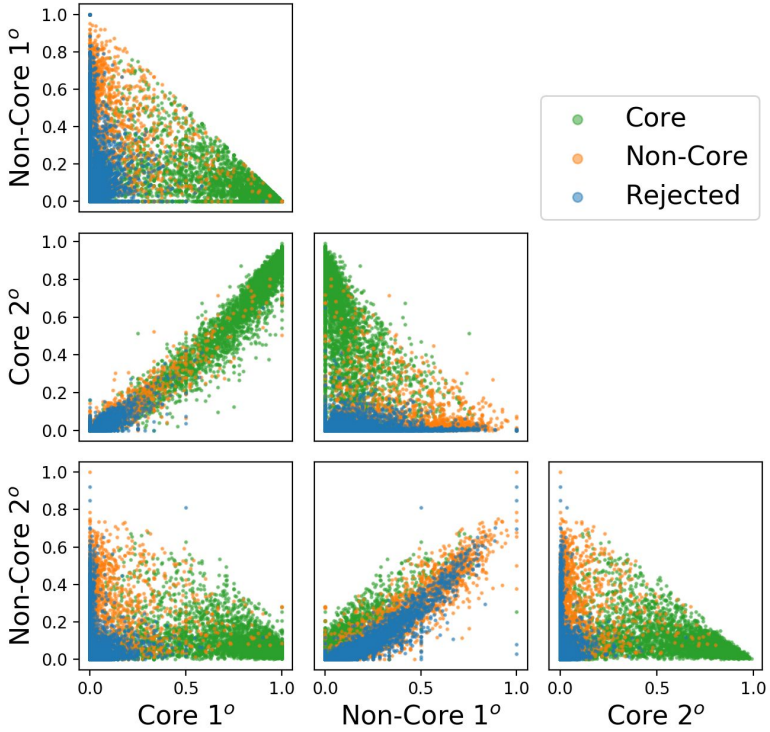
$$f_{Core} = \frac{A}{N} \qquad f_{NonCore} = \frac{B}{N}$$

- Core vs. rest well separated.
- Problems for references not yet in INSPIRE.



References of references

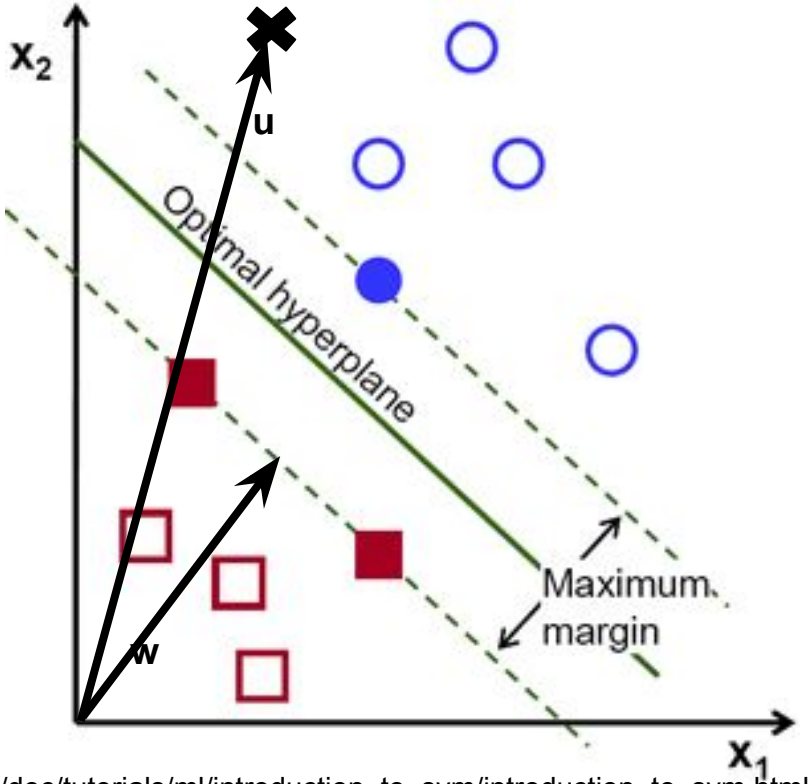
- Look at the references of each reference
- Calculate the fraction of Core, Non-Core for a “second-order” estimate of reference fractions.
- Scatter each of these.



Support Vector Machine (SVM)

How does an SVM classify points?

- **w** -vector perpendicular to optimal hyperplane
- **u** -unknown vector
- Condition: **w** · **u** +b ≥ 0
- If true ○
- If false □

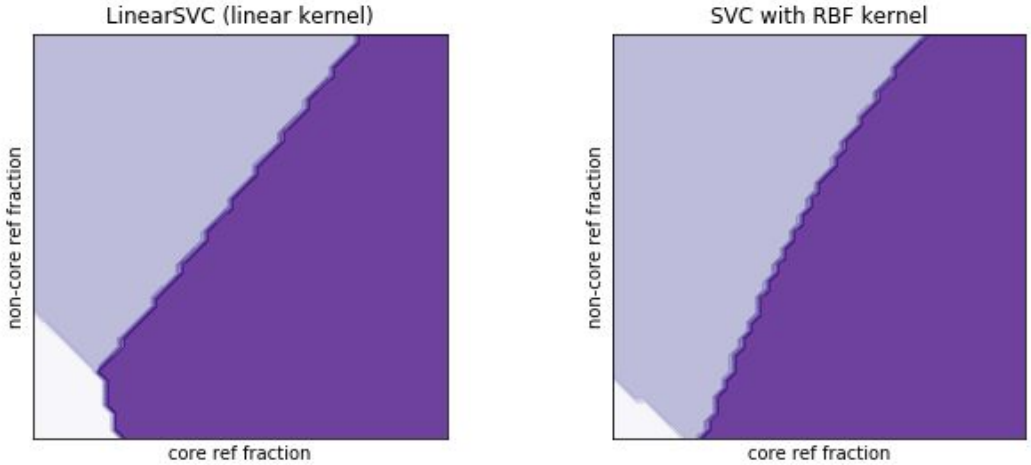


https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Support Vector Machine (SVM)

Our problem

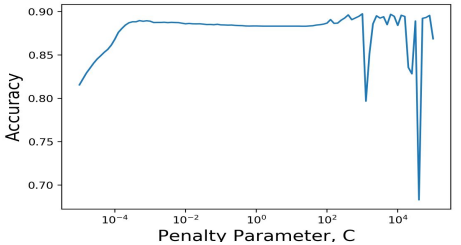
- 4 features - hyperplane
- Multi classification- One Vs Rest

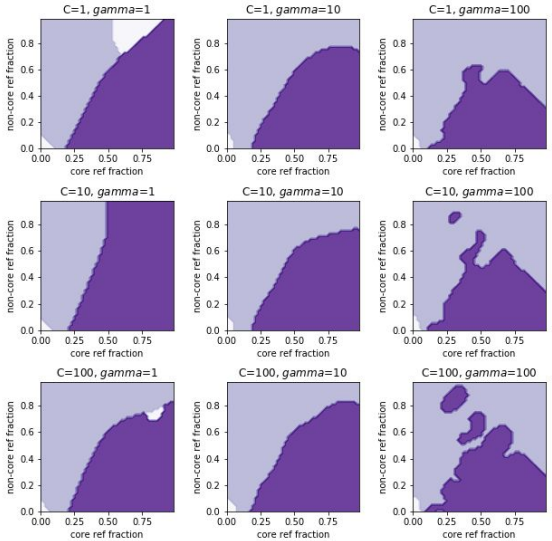


Support Vector Machine (SVM)

Optimising hyperparameters

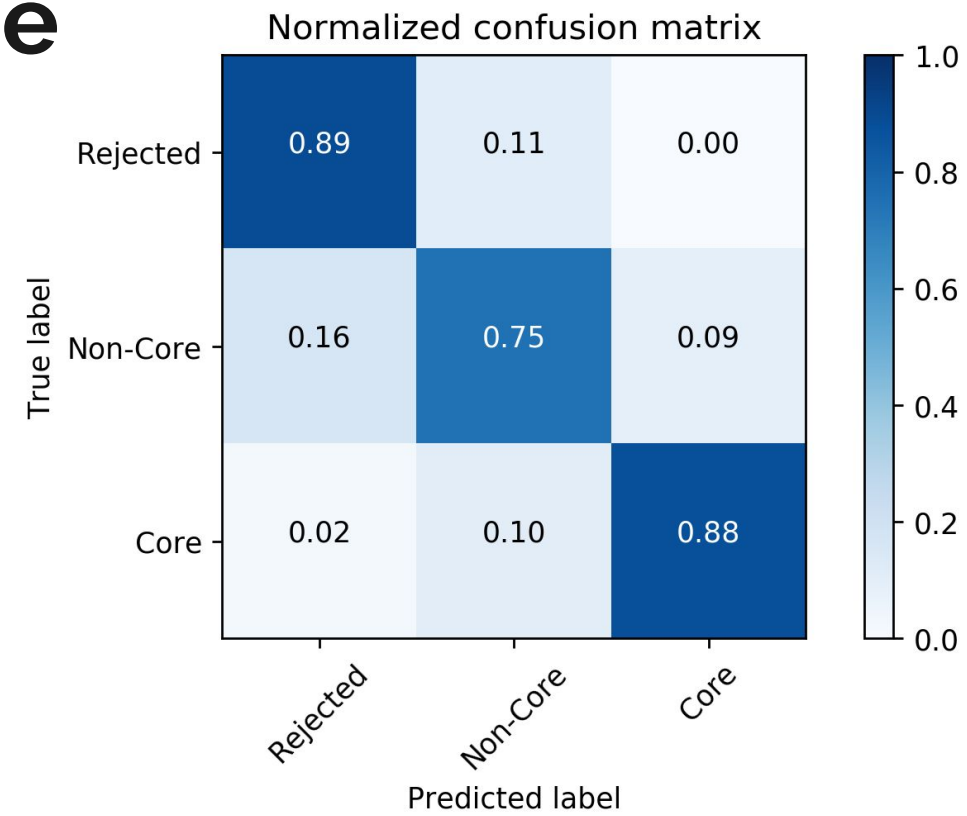
- Kernel: Linear vs. RBF (radial basis function)
- Penalty Parameter, *C*.
  - High C: tries to classify all training points correctly - overfitting
  - Low C: allows misclassifications- better generalisation
- Reach of single training example, γ
  - High γ: only uses points close to decision boundary as support vectors
  - Low γ: support vectors have greater sphere of influence





SVM Performance

- Linear SVM with four features (reference fractions), with C = 1.0.
- Lower overall accuracy than ‘Bag of Words’ or ‘Word embeddings’
- ...but significantly fewer false negatives



Accuracy: 0.88

Combining methods

- Use NN output class weights as input features into SVM
- SVM features:
  - Core weight
  - Non-core weight
  - Rejected weight
  - Core refs
  - Non-core refs
- Not worth it?

Automating the decision

- If the SVM tells us if a paper is Rejected, Non-Core or Core can we then determine if we should trust it?
- If the SVM is sure it is correct we won't need a person to check the decision.
- The distance each entry is from the decision boundary can be used to find this out.

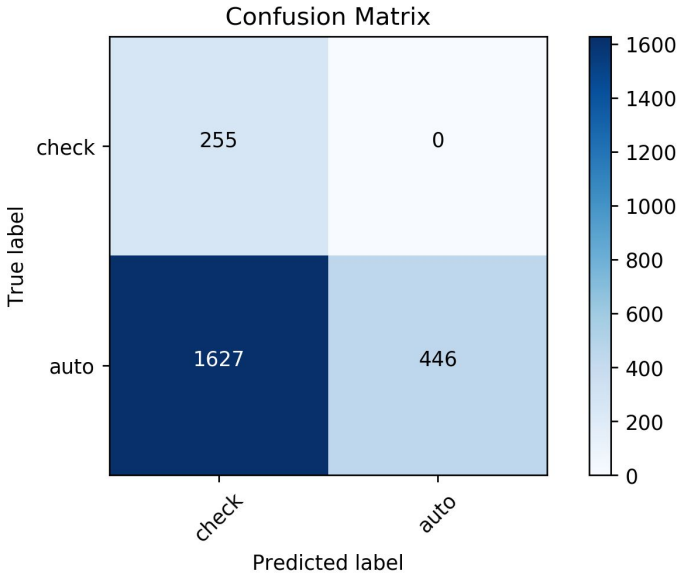
- Looking at what unites the misclassified entries may give insight into what else could be used features to help tell them apart

Two Classifiers

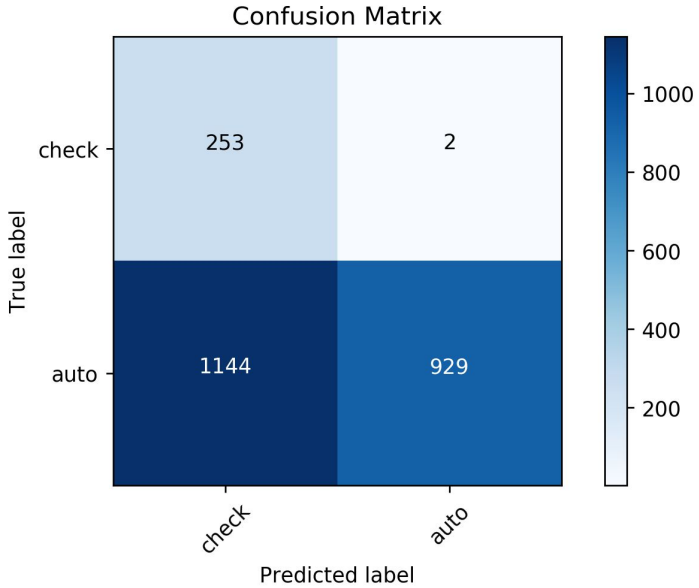
- Can you use a second classifier to see if you need to check results from a first SVM?
- The first SVM would use the 4 fractions of references as input to determine if a paper was Rejected, Non-Core or Core.
- The second would use the result from the first SVM (ie, the distance each point is from the decision boundary) and a selection of other features (Number or references, Dictionary of Keywords, Category, etc) to deduce if you need to check the result of the first.

Two Classifier Results

- features: decision distance, number of keywords, the predicted decision and category



- features: decision distance, number of keywords, the predicted decision and category, number of refs



Problems with this approach

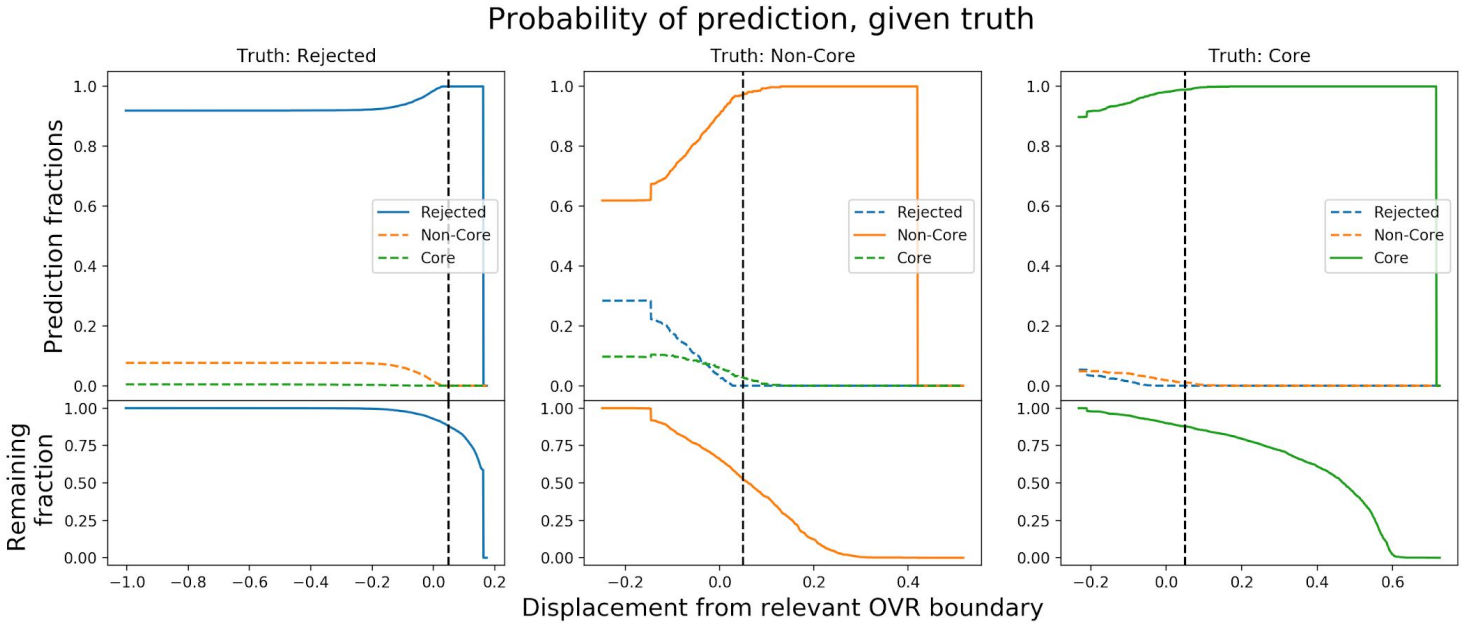
- Has become quite complicated
- It requires you to extract many features, rather than just references
- You train the 2nd classifier on the validation data from the first. This means training set is significantly smaller than the 1st classifiers.

Distance from decision boundary

- Main requirement is to reduce false negatives (ie, rejecting a Core/Non-Core paper is very bad).
- One vs the rest (OVR) gives 3 distances per point:
  - Distance from REJECTED vs rest boundary
  - Distance from NON-CORE vs rest boundary
  - Distance from CORE vs rest boundary
- Normalise by feature weights for each OVR classification.

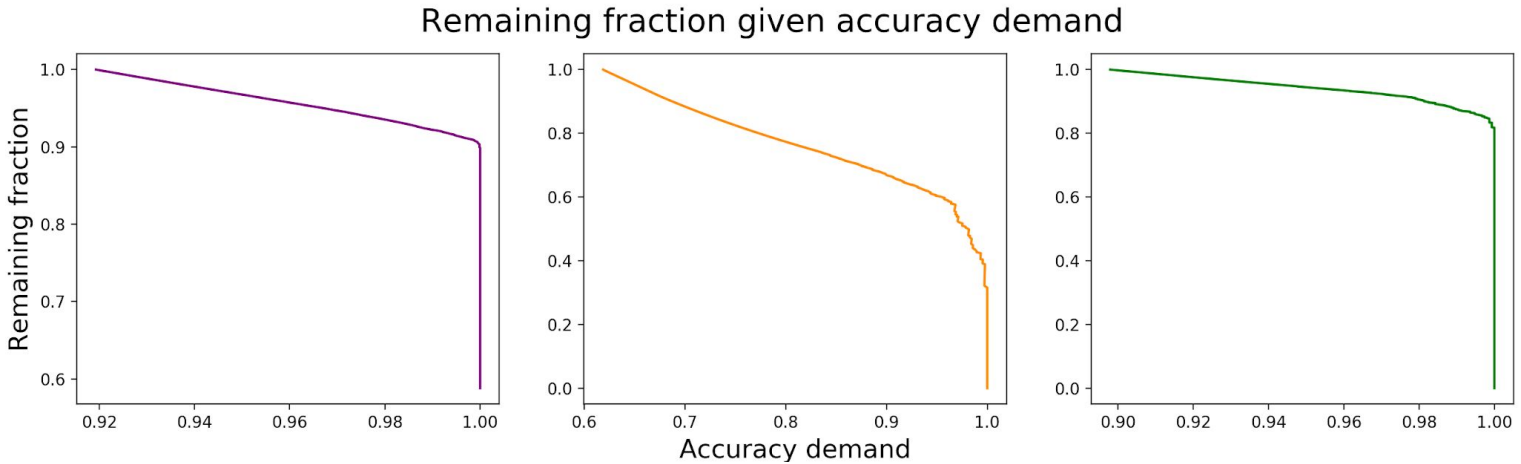
P(Prediction|Truth)

- If we select all papers greater than a certain distance from the boundary, what is the probability of a prediction, given the true label?

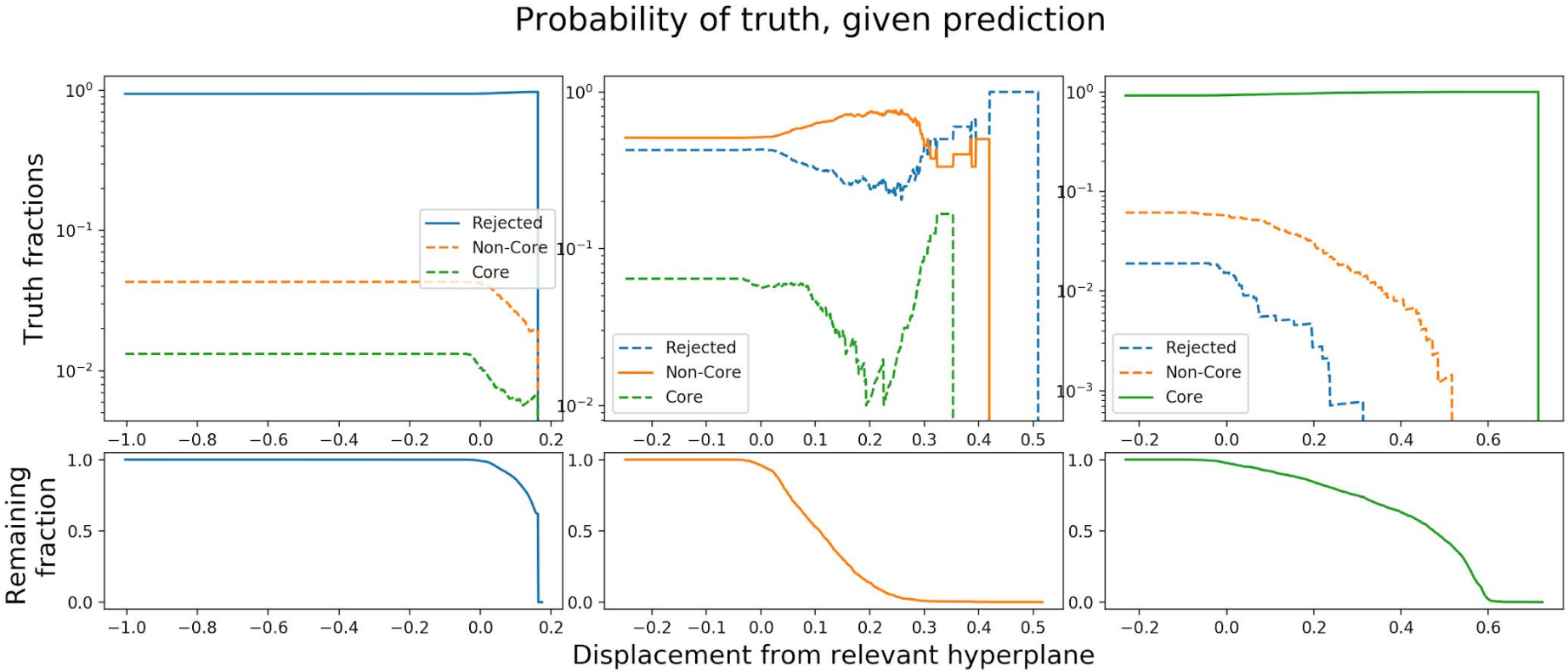


P(Prediction|Truth)

- Can also determine what fraction of papers will remain, if given an accuracy demand.

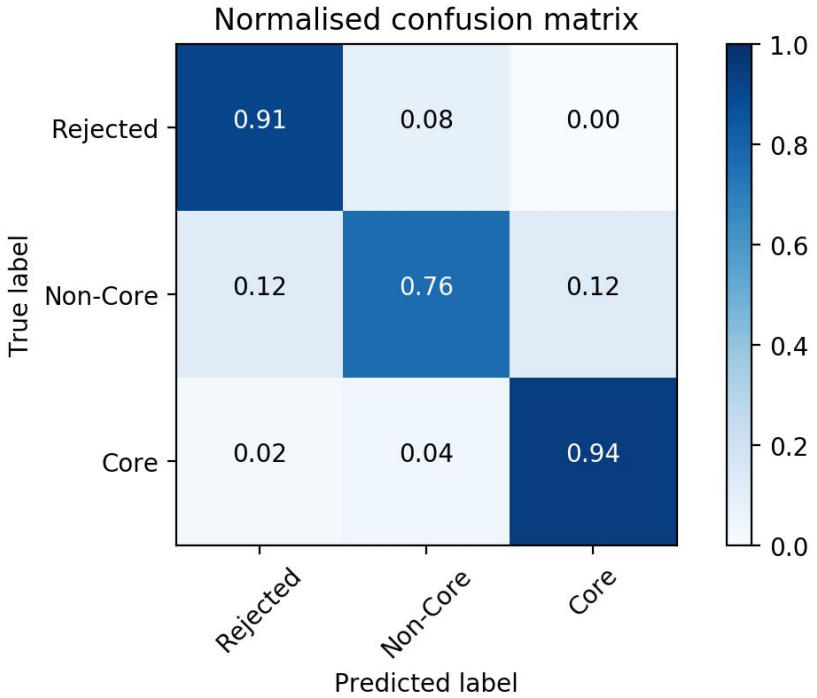


P(Truth|Prediction)



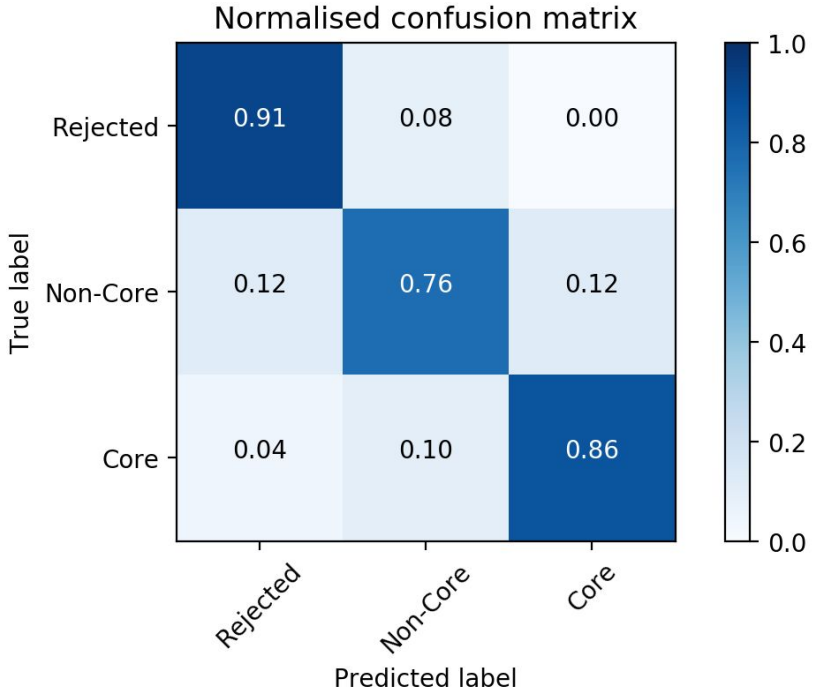
Performance after distance cut

- Confusion matrix using data points with:
  - Rejected: >0.16
  - Non-core: >0.05
  - Core: >0.05
- 69% of data would be automatically classified



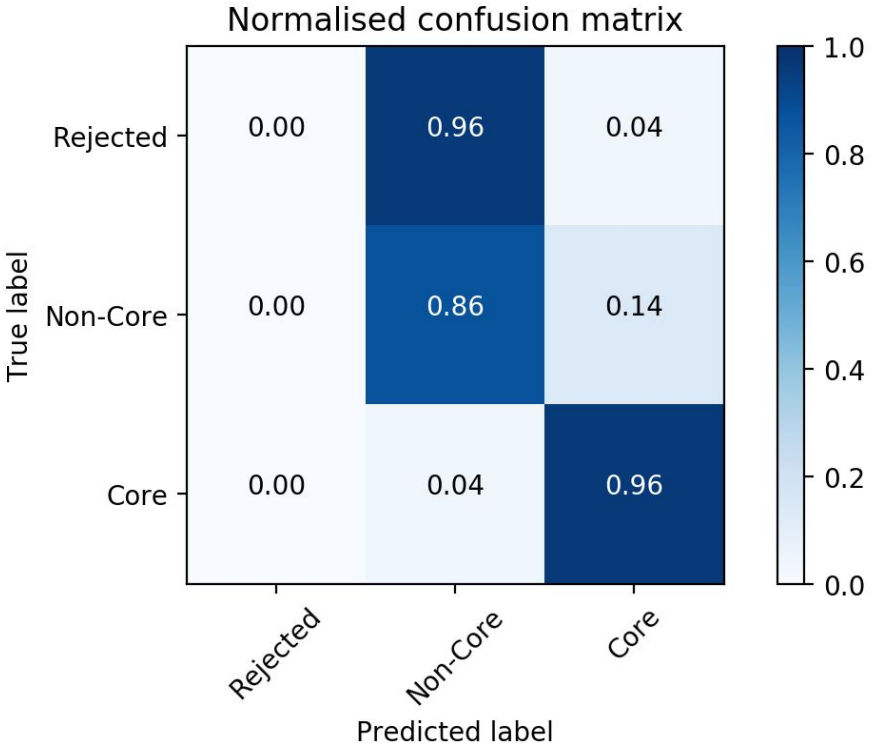
Applying cut off/excluding HEP

- arXiv categories automatically classified as CORE:
  - hep-ex
  - hep-lat
  - hep-ph
  - hep-th
- How do we do after we take away these categories?
- 67% data set would be automatically classified excluding HEP

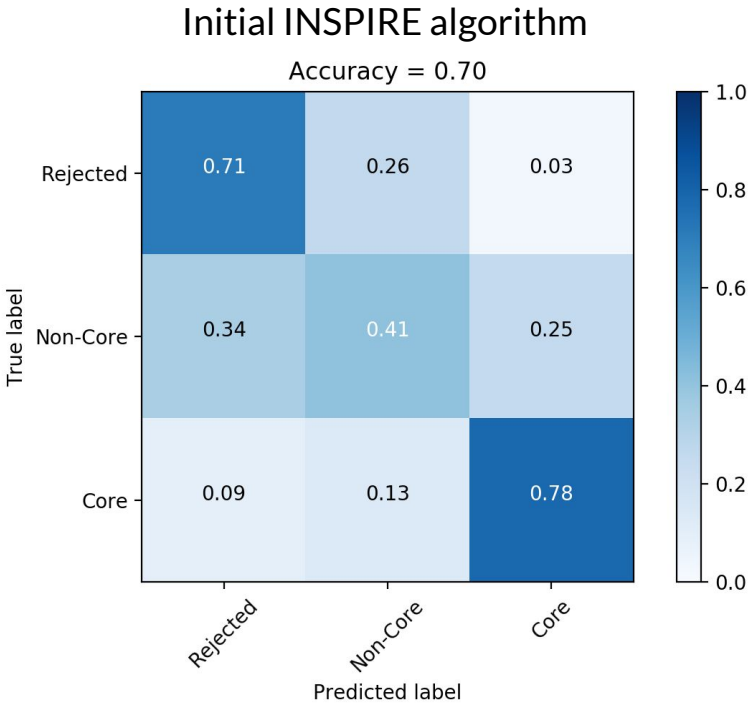


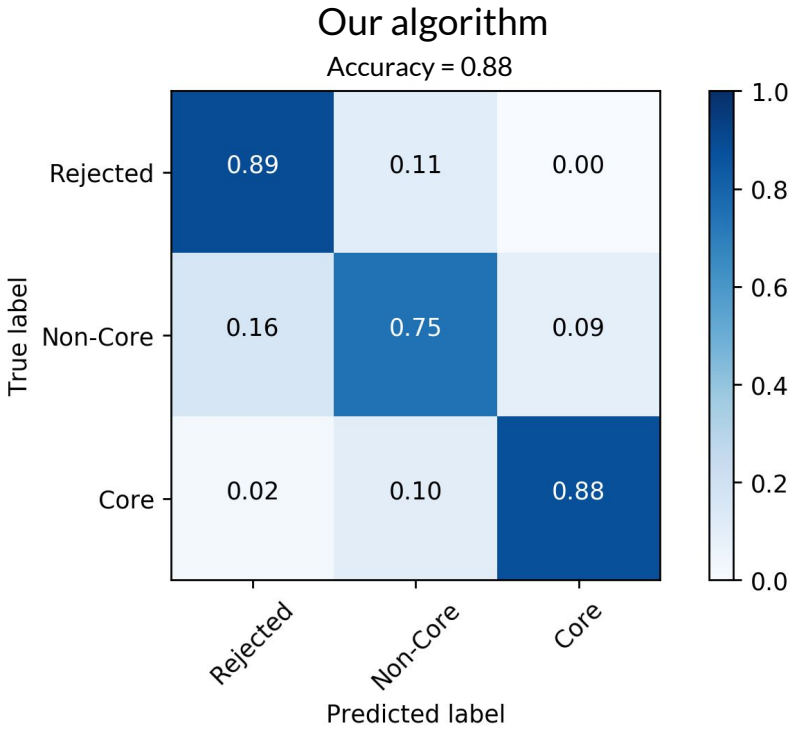
After requiring no false rejections

- What if we are even more restrictive with false Rejected classifications?
- 26% data set classified automatically with no false rejections
- 17% excluding hep



Did we improve?





Further work

- Two binary classifiers, Rejected vs. Accepted, then try to classify accepted into Core/Non-Core.
- Multiple classifiers based on category.
- Use an unbalanced loss function to penalise false rejections more
- Try to estimate P(Prediction|Truth) for an unseen paper using Bayes' rule.

$$P(P|T) = P(T|P)*P(P)/P(T).$$

Conclusions

- A simple SVM performs as well for our requirements as more complex models
- Non-Core proves hard to classify.
- It is possible to automate classifying papers without falsely rejecting them for around a quarter of the dataset.

Thanks!

Back up slides

Support Vector Machine (SVM)

How do we find the optimal hyperplane?

- $\mathbf{w} \cdot \mathbf{x}_+ + b \geq 1$
- $\mathbf{w} \cdot \mathbf{x}_- + b \leq -1$
- Maximize the margin subject to these constraints to find optimal **w** and *b*.