Machine learning for classification with big data in price statistics production pipelines

Edward Rowland

Hazel Martindale

Prices Alternative Data Sources Project – UK Office for National Statistics

Contact: edward.rowland@ons.gov.uk

Talk outline

PADSP – Building a pipeline

Pre processing → Classification → Averaging → Outlier detection → Imputation → Retailer indices → Index aggregation

Machine learning (ML) approaches to classify price quotes from web-scraped data to COICOP5/ONS item level

Go over problems and (possible) solutions to each

Paper - describes a complex clothing classifier in more detail

Talk structure

Problem 1 – Making sense of text data

Problem 2 – Lots of data, few labels

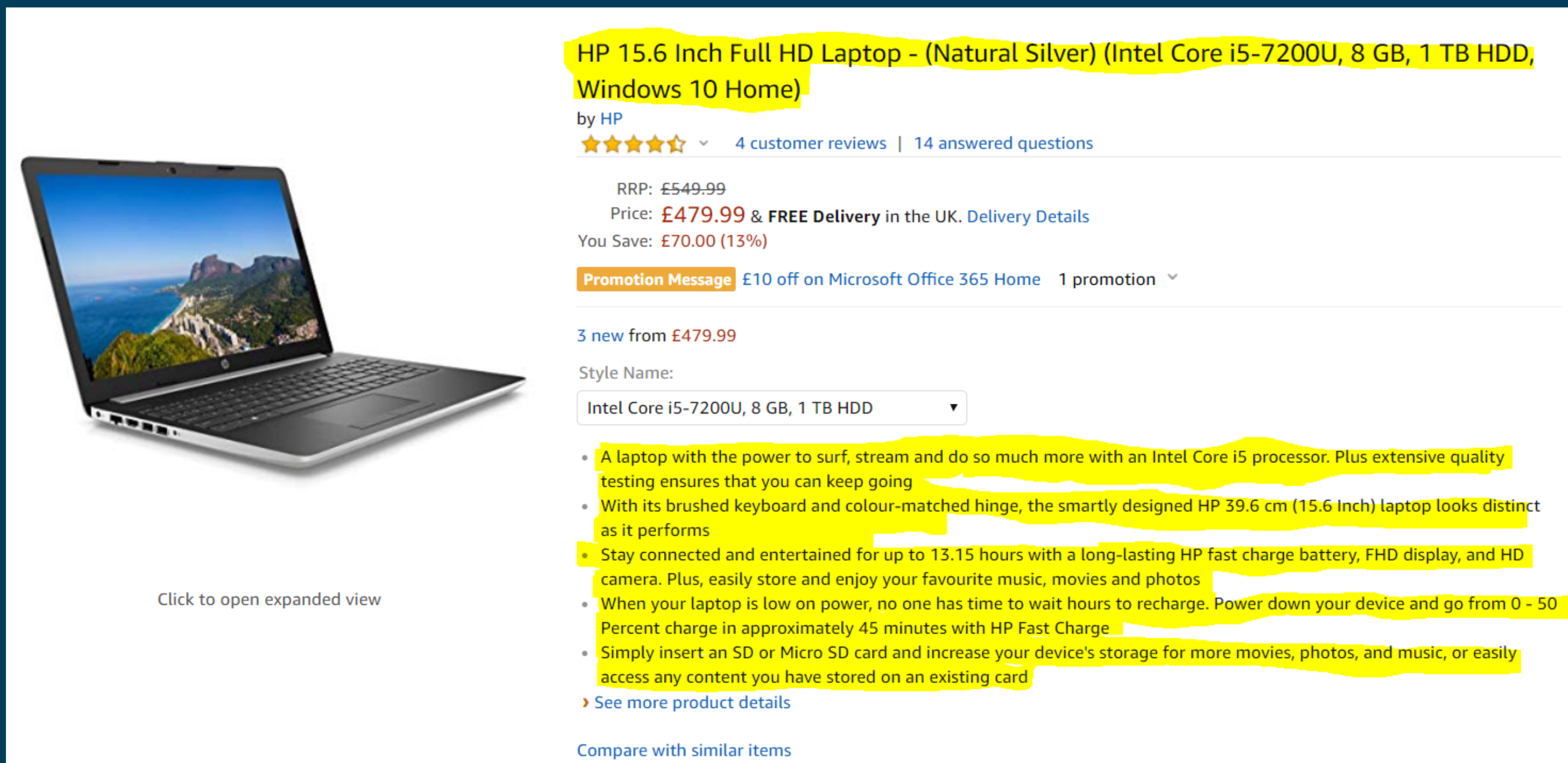
Putting this all together – an example clothing classifier

Problem 3 (If time permits) – Is it good? Measuring classifier performance

Problem 1

MAKING SENSE OF TEXT DATA

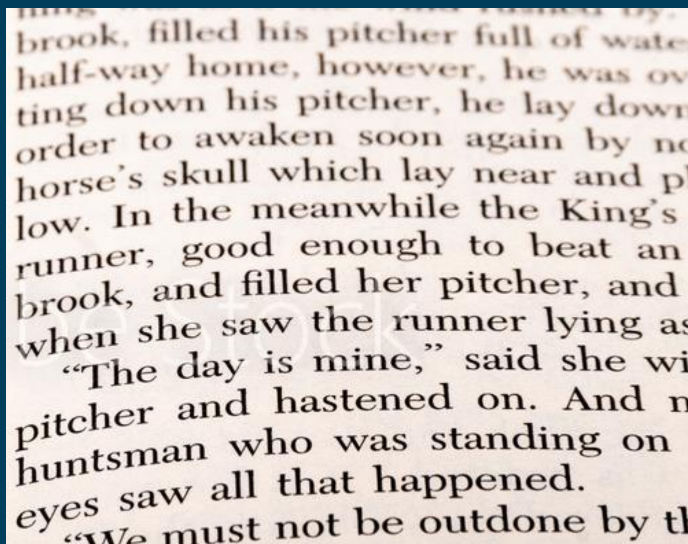
The problem



How do we make use of the text in machine learning algorithms?

Word embedding

A family of Natural Language Processing (NLP) approaches to turn text into numerical data


brook, filled his pitcher full of wate
half-way home, however, he was ov
ting down his pitcher, he lay down
order to awaken soon again by no
horse's skull which lay near and pl
low. In the meanwhile the King's
runner, good enough to beat an
brook, and filled her pitcher, and
when she saw the runner lying as
"The day is mine," said she wi
pitcher and hastened on. And n
huntsman who was standing on
eyes saw all that happened.
"We must not be outdone by th

Hopefully, this provides useful features in a classification algorithm. Lets go through a few…

Count vectorisation

Counts occurrences of words in sentence, e.g.

Sentence1: “There is a black cat”,

Sentence2: “A black cat and a black ball”,

Sentence3: “Is there a black cat?”

Note that 1 and 2 have the same coding, bi-grams can help with this

	black	there	is	cat	and	ball	a	‘is there’	‘there is’
Sentence 1	1	1	1	1	0	0	1	0	1
Sentence 2	2	0	0	1	1	1	2	0	0
Sentence 3	1	1	1	1	0	0	1	1	0

Term Frequency – Inverse document frequency (TF-IDF)

TF – how often a word (t) appears in a document containing N words

$$TF(t) = n_t/N$$

IDF – how few documents contain a word(t) (df(d,t)) of all docs in corpus (C)

$$IDF(t) = \log \frac{C}{df(d,t)} + 1$$

For each word in each document, then multiply

$$TFIDF = TF * IDF$$

Assumes less common words have more task relevance

word2vec

Two layer neural network trained on a corpus to predict the next word in a sentence

e.g. “The mouse eats cheese from the box”

Target word : cheese

The network aims to predict the target word, given the preceding words

We take the weights of the last layer in the neural net as the vector for the target word

fastText

Similar to word2vec, except can predict out-of-corpus words

Breaks down words into character n-grams e.g. apple would have ap, pp, pl, le bi-grams.

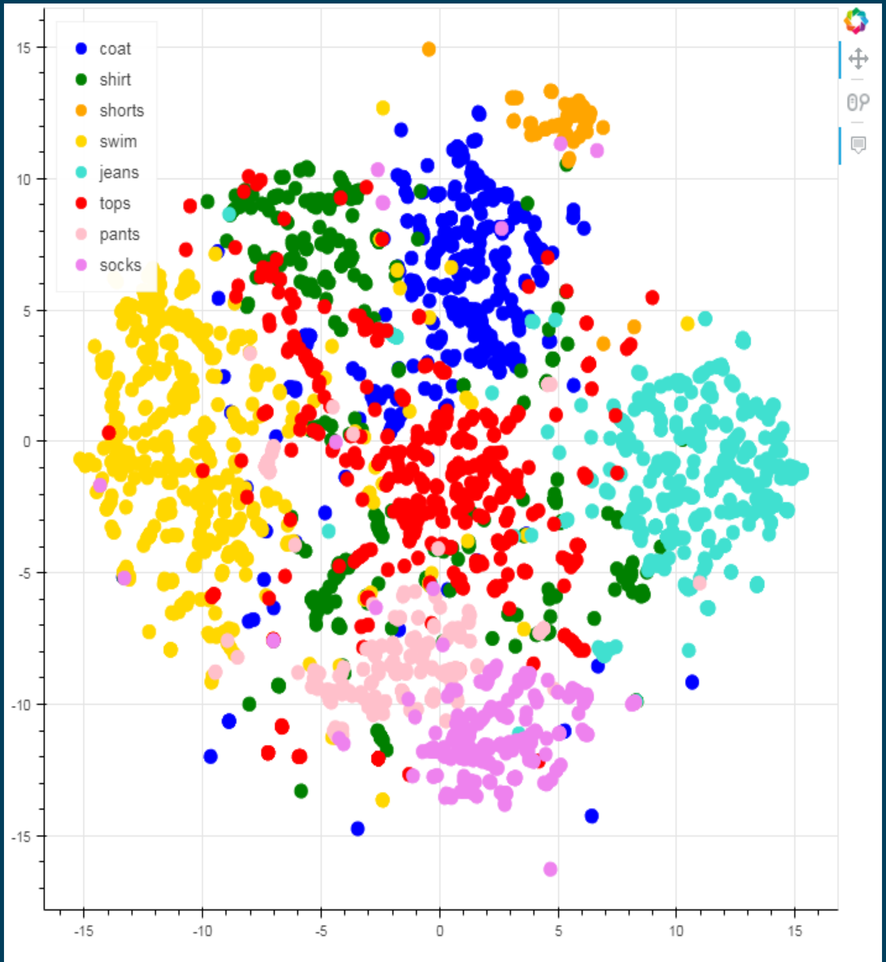
Then predicts the next bi-gram in a sentence.

Therefore, it works with words it has not been trained on

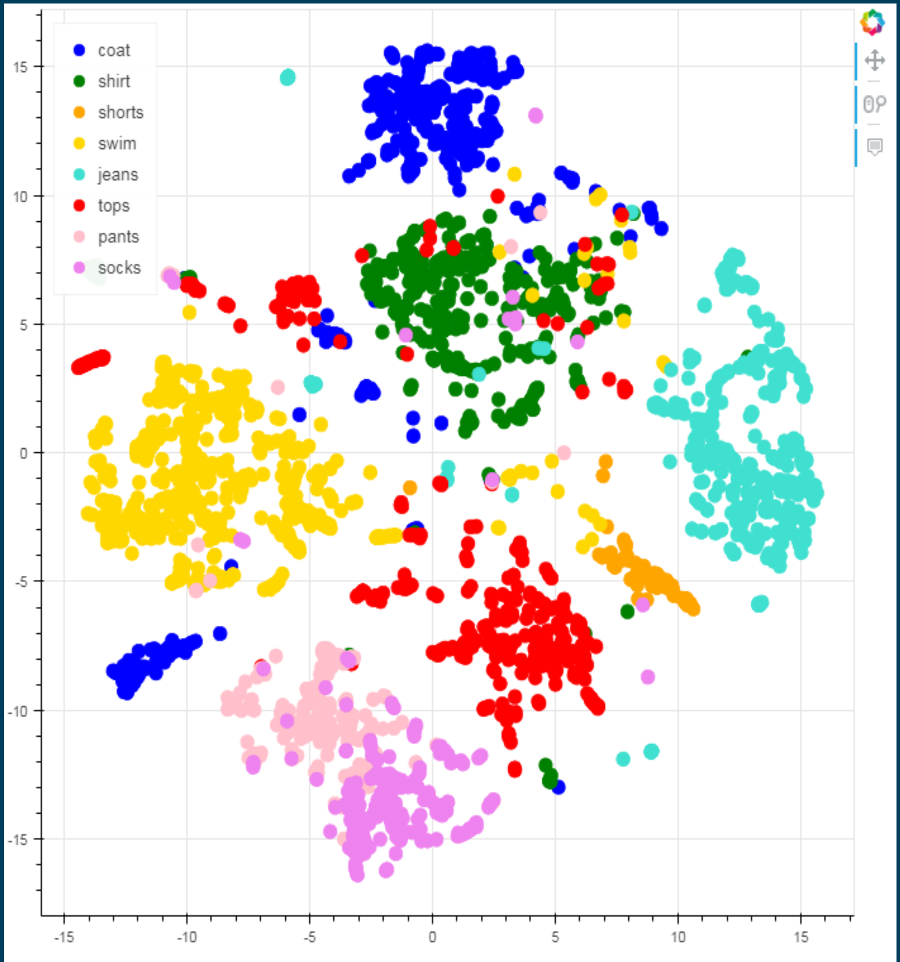
What do they look like

From word embeddings for garment names in clothing data, 2D t-SNE projection

TF-IDF



word2vec



Take home messages

Can help you out, if the data looks ‘clustery’

Lots of ways of solving this problem

- Count vectorisation
- TF-IDF
- word2vec
- fastText

And some more advanced solutions not covered here

- BERT
- ELMO

Problem 2

LOTS OF DATA, FEW LABELS

The problem

Supervised classification algorithms need to be told what to do before they work out how to classify new data.

This requires labelled data to train and evaluate the approach(es)

Meaning somebody has to accurately assign COICOP5 labels to your data

Very time consuming with tens of thousands or millions of observations, how can we avoid this?

Fuzzy matching

We compute measures of similarity and assign label to those above a certain similarity score

Levenshtein/edit distance – number of insertions, deletions or substitutions

- Sensitive to string length and word order

Partial ratio – similar levenshtein, but matches substrings

- Insensitive to string length and word order

Jaccard distance – intersection/union

- Sensitive to string length, insensitive to word order

Fuzzy matching

<table>
<tr><th>Comparison</th><th>Edit</th><th>Partial ratio</th><th>Jaccard</th></tr>
<tr><td>blue men’s shirt</td><td rowspan="2">4</td><td rowspan="2">100%</td><td rowspan="2">0.89</td></tr>
<tr><td>navy blue men’s shirt</td></tr>
</table>

Good where similar text strings should have the same label

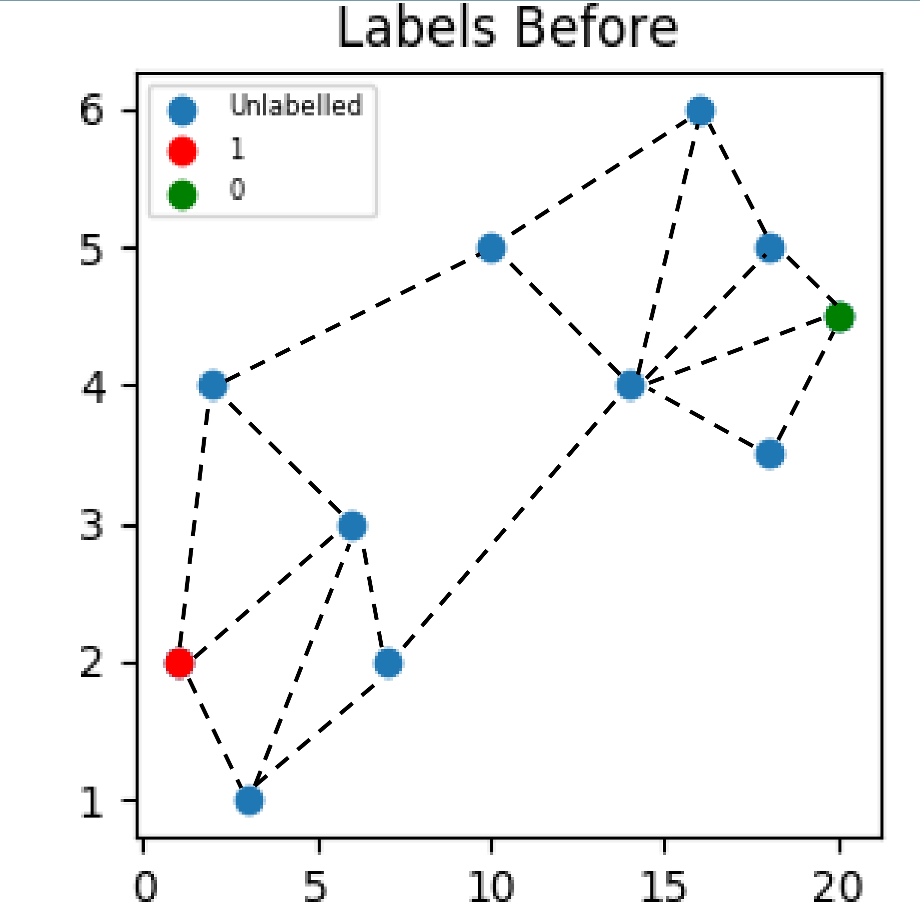
but probably is not going to label everything and likely domain specific.

These methods, Levenshtein especially, scale poorly with increasing data.

Can mitigate by capping edit metric, e.g. 0, 1, 2, >2.

Label propagation

- Semi-supervised method to increase labels
- Uses a small number of labels
- Represent entire dataset in feature space
- Construct graph from data structure (usually K-NN)



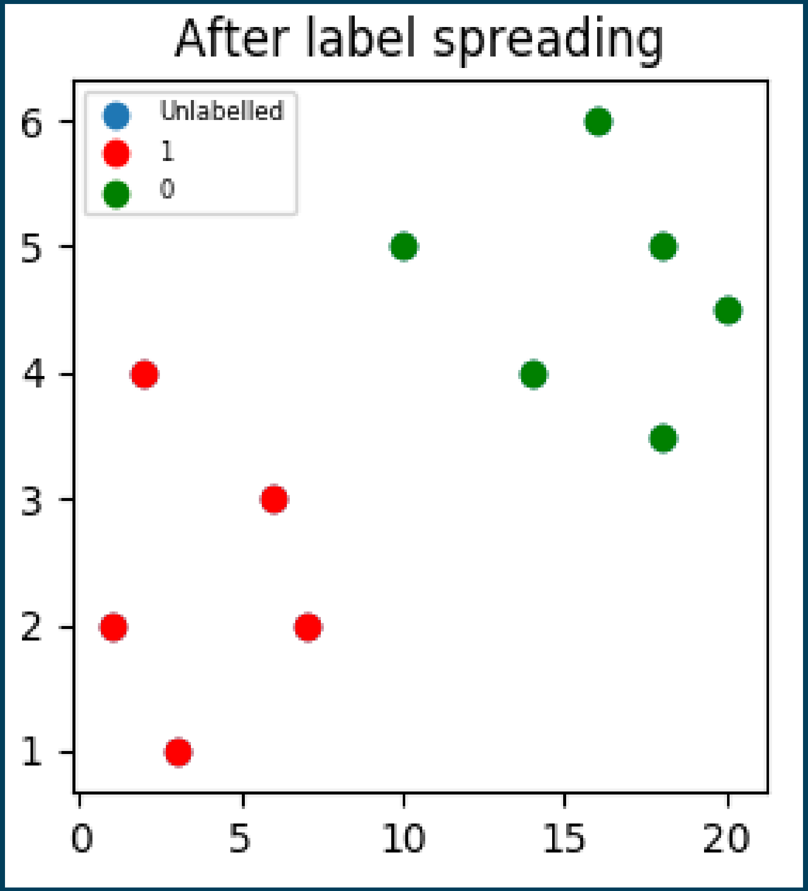
Label propagation

Probability of an unlabelled point being assign each label are calculated by Weighted distance along edges connected to labelled points

Highest probability label assigned to each point, then repeat

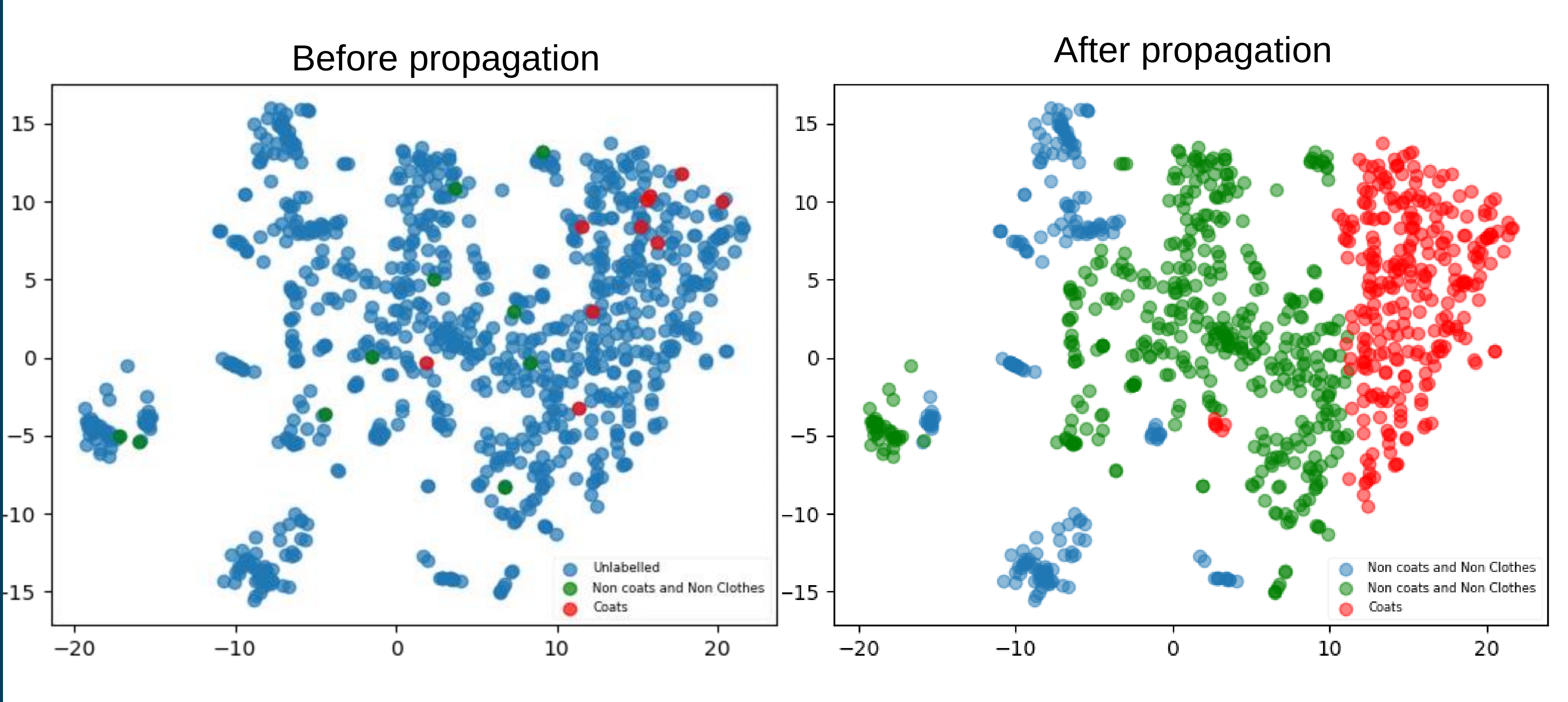
How the graph is built has big effect on the final labels

Label spreading variant allows for noise in assigned labels

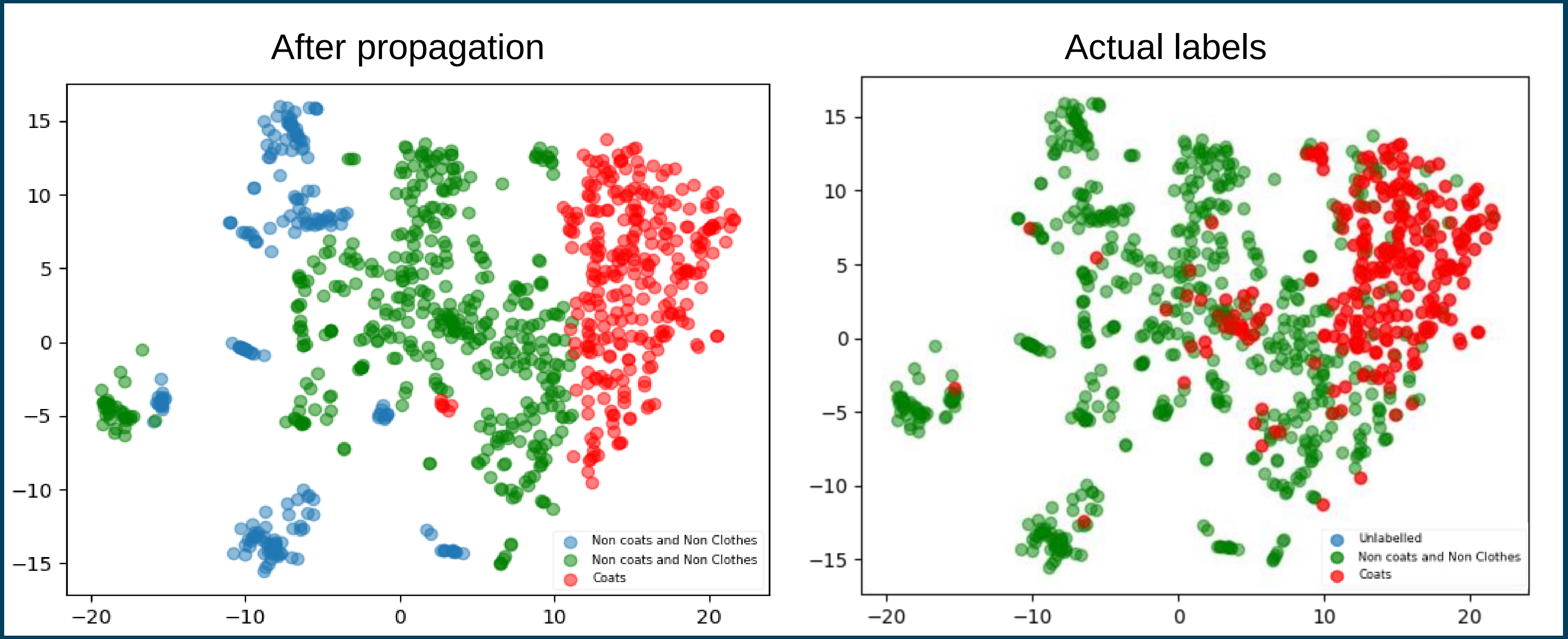


Label propagation

Example with clothing product names



Label propagation



Label Propagation

Can work well if…

- Your data has a ‘clustery’ structure in feature space,
- the structure relates to your labels
- and the initial ‘seed’ labels are representative.

Some issues…

‘Junk’ items are particularly hard to find representative labels for

There are unresolved issues with scaling as

K-NN does not distribute effectively

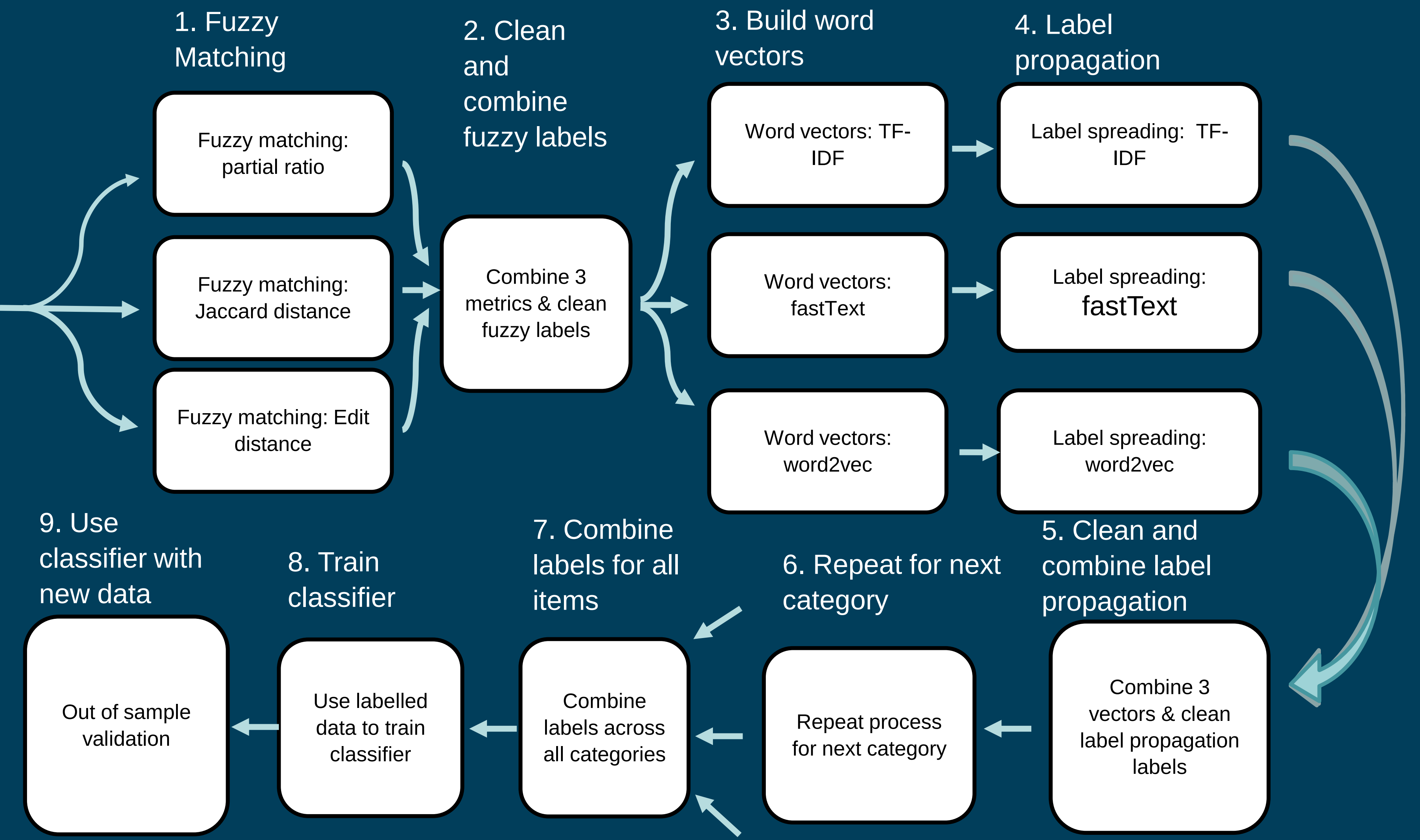
PUTTING THIS ALL TOGETHER

WGSN dataset

- Used sample from WGSN data
- Split into two, to train and evaluate
- Each half contains 3485 observations
- Very few initial labels
- Label propagate and train first half
- Second classified compared against manual labels

Item Description	Unique products
‘Junk’ Items	981
Women's Coat (SEASONAL)	430
Men's Casual Shirt, long or short sleeved	379
Women's Sportswear Shorts	65
Women's Swimwear	593
Boy's Jeans (5-15 Years)	512
Girl's Fashion Top (12-13 years)	411
Men's Pants/Boxer Shorts	235
Men's Socks	239

Overall method for clothing data



Results

Expand 91 original labels to 1692 (48.5%) labels with fuzzy matching

Then expand these to 2802 (73%) with label propagation

And then classify, giving the below results

	Training data			Test data		
Metric	Precision	Recall	F1-score	Precision	Recall	F1-score
Non-linear SVM	0.89	0.89	0.89	0.85	0.83	0.84
Decision Tree	0.88	0.88	0.88	0.84	0.85	0.83
Random Forest	0.88	0.88	0.88	0.86	0.86	0.86

Caveats

Toy data – real data likely to be more unbalanced

Some cleaning for sense checking applied e.g. item with “girl’s” in a men’s category is removed

Performance across classes is not uniform

- Men’s casual shirts; hard to distinguish from formal
- Woman’s swimwear; separates incorrectly classified
- Woman’s sportswear shorts; small sample
- Woman’s coats; similar items in ‘junk’ e.g. girl’s coats

Conclusions

Despite issues,

- Classifying nearly 4000 clothing items
- with a F1-score of 0.86
- Using only a short text description
- and 91 labels

Is extremely promising

Future work

More data!

- 10k labels ~80,000 unique items
- Enough data to build price indices from

Better classifier!

- Hyper parameter tuning
- Pretrained word2vec and fastText on large (e.g. Wikipedia) corpus

Dealing with unwanted items better

– positive unlabelled learning

Problem 3

IS IT GOOD? MEASURING CLASSIFIER PERFORMANCE

Measuring classifier performance

The classification output is not the end product, the price index is

Classifier performance should be considered in this context.

How might it affect the

- Variance
- Bias

of the price index?

Unbalanced data

Surely you just look at how often it is correct...?

Consider a dataset where 90% of data should be included in the index

A naïve classifier would include everything (label all as true). 90% accuracy, Great

Actually, it always makes false positive errors! Therefore we need to consider

- True positive rates (recall)
- False positive rates
- Precision

What about the impact of different errors?

A doctor sees a patient with symptoms that might be a serious condition or a minor one.

Do they refer them to a specialist even if it is much more likely to be a minor condition?

Yes – the risk of a false positive (sending them to a specialist) is lower than a false negative.

The same is for price indices - we need to consider the impact of

- false negatives (incorrectly excluded)
- false positives (incorrectly included)

Metrics – define terms

- **True positive** – *correctly included*
- **True negative** – *correctly excluded*
- **False positive** – *incorrectly included*
- **False negative** –*incorrectly excluded*

		Classified		Truth
		Positive	Negative	
Label	Positive	True +ve	False -ve (Type II)	Total labelled +ve
	Negative	False +ve (type I)	True -ve	Total labelled -ve
Classified as		Total classified +ve	Total classified -ve	Total observations

Some ways of measuring classifier performance

Balanced accuracy/average recall

- $recall = \frac{True\ positive}{True\ Postive + False\ Negative}$
- Mean recall for all classes

Fβ-score

- $Precision = \frac{True\ positive}{True\ positive + False\ positive}$
- $\mathrm{F}\beta = (1+\beta^2) * \left(\frac{precision * recall}{(\beta^2 * precision) + recall}\right)$
- β determines importance of recall over precision
- β = 1 equal importance for both

Multiclass classification

Reduce this to a binary problem via one-versus-rest for each class

- True labels; a single class
- False labels; everything else

Micro average

- Sum quadrants (TP, TN, FP, FN) for each 1-v-rest
- Compute metrics from summed values

Macro average

- Compute metrics from all 1-v-rest
- Compute (weighted) mean of these

Multiclass classification

Reduce this to a binary problem via one-versus-rest for each class

- True labels; a single class
- False labels; everything else

Micro average

- Sum quadrants (TP, TN, FP, FN) for each 1-v-rest
- Compute metrics from summed values

Macro average

- Compute metrics from all 1-v-rest
- Compute (weighted) mean of these

When to use?

Micro

- does not take into account individual class performance
- All observations are equal
- Insensitive to class imbalance

Macro

- can examine individual class performance
- Over-represents small classes
- Can correct with weighting, but will this change in future?

Current and future work

Initial guidelines on metric usage – email (Edward.Rowland@ons.gov.uk) for a copy

Future work

Develop a quality framework for classification

- Relationship between index quality and metrics
- Guidelines on quality control and checking
- Expenditure weights and cost functions
- How often to retrain classifiers? Batch or active?

THANK YOU FOR LISTENING