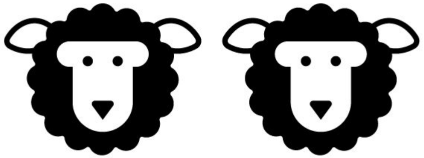


AVITO DUPLICATE ADS DETECTION

Alexey Grigorev
Team **ololobhi** (Abhishek & ololo)

Data set

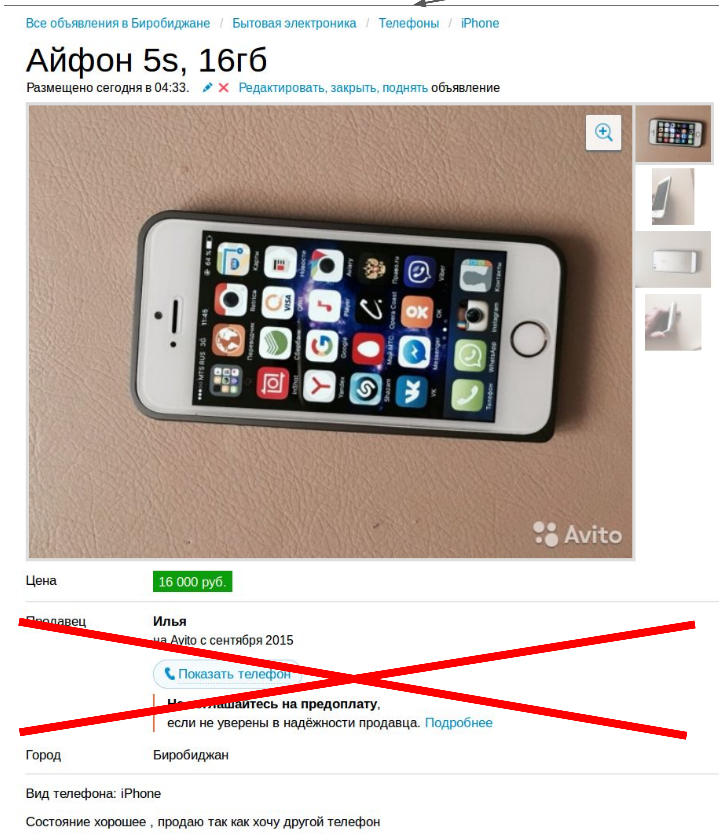
- ~3 mln train pairs, ~1 mln test pairs
- ~10.8 mln images (~45 gb)

Target

	title_1	title_2	price_1	price_2	isDuplicate
0	Продаю телефон	Продаю телефон Samsung Galaxy J1	6500	6000	1
1	Б-м-в	Оригинальные диски бмв	23500	23500	1
2	Balmain, Burberry, prada, Gucci, Armani	Dsquared2, Dsquared, Burberry, balmain, Gucci	49900	60000	0
3	Обшивки задние ВАЗ 2110-12	Обшивки ВАЗ 2110	1650	3000	0
4	Chevrolet Aveo, 2011	Chevrolet Aveo, 2011	278000	285000	1
5	Коллекционная гитара Guyatone made in Japan	Mustang Tomson Splender Series made in Japan	15000	15000	0
6	Комфортабельные Грузоперевозки	Комфартабельные Грузоперевозки	150	150	1
7	Gta 5 ps3	Gta5 для ps3	1000	1500	1
8	Деревянные кубики	Азбука в кубиках	350	500	0
9	Кроссовки Air Max 2015 Nike	Nike air force 1	3000	3590	0

Evaluation metric: AUC

Category_ID



Title

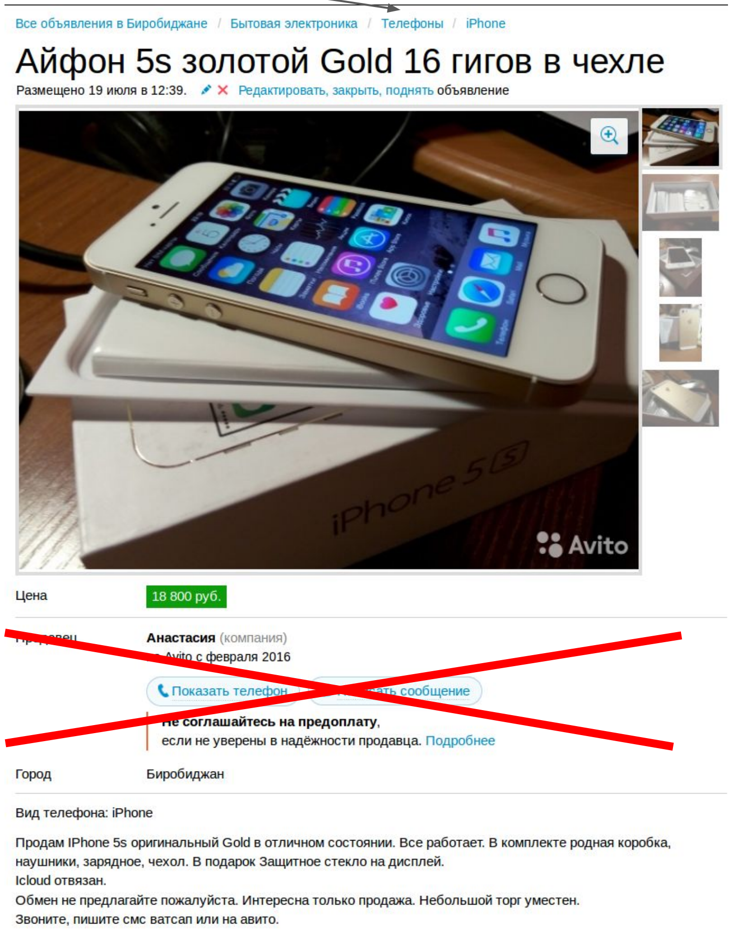
Pictures

Price

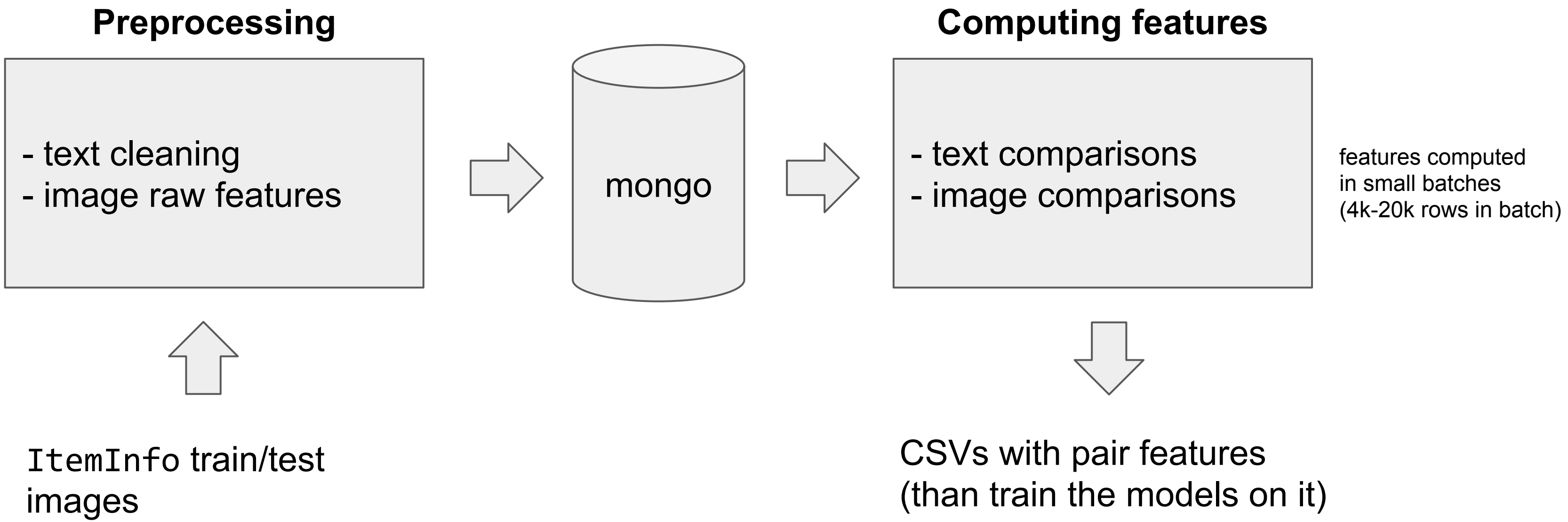
No seller data

locationID
attrsJSON

Description

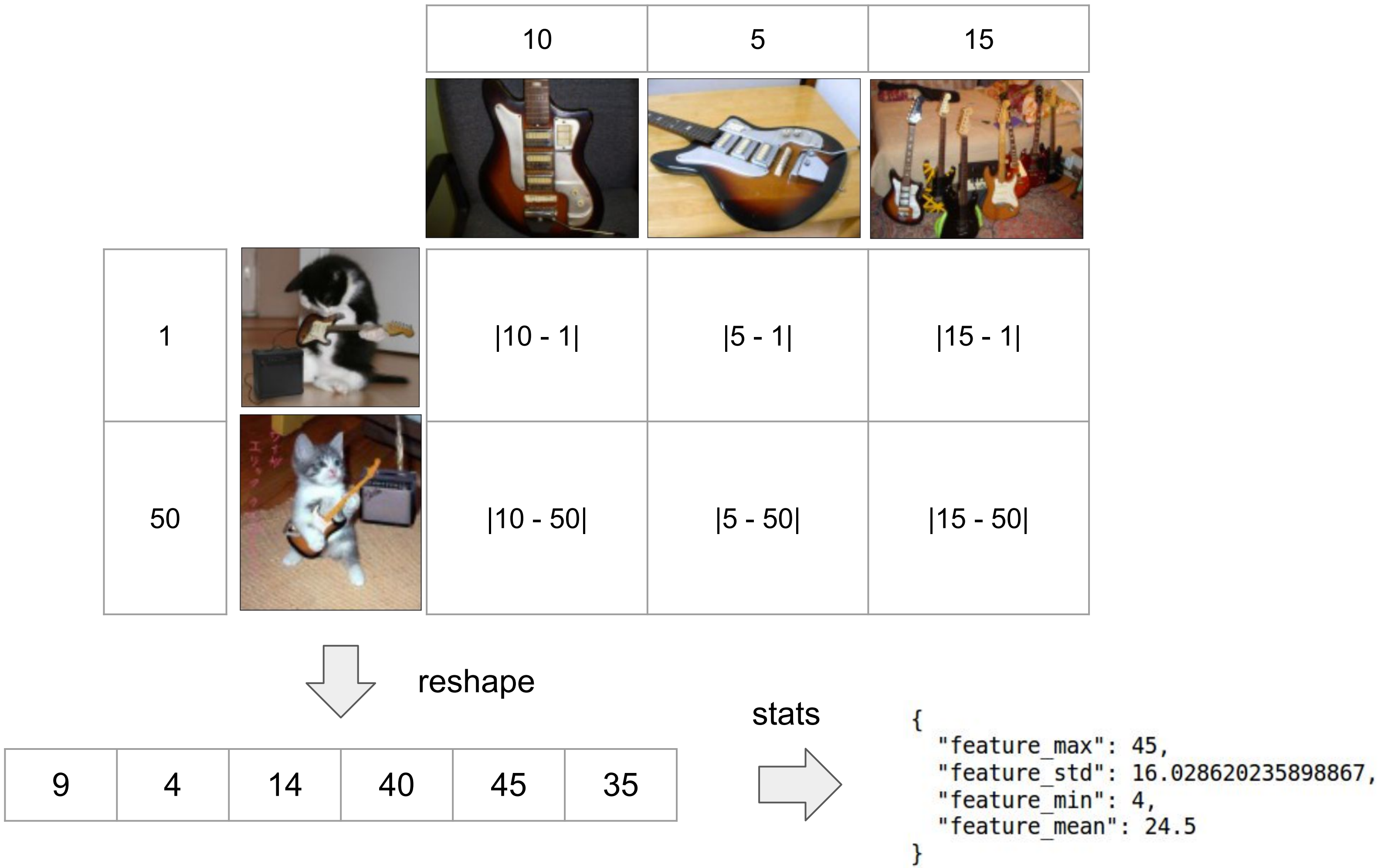


Process Overview



Features 1

- Simple Features
  - CategoryID (plain, no OHE)
  - Number of images
  - Absolute price difference
- Simple Text Features
  - Num of Rus/Eng/Digits chars
  - Length of Title, Description
  - 2-4 ngram similarity on char level
  - Fuzzy string matches (via FuzzyWuzzy)
- Simple Picture Features
  - Channel statistics (min, mean, max, etc)
  - File size differences
  - Geometry matches
  - Num of exact matches via md5 hash
- Simple GEO Features
  - MetroID
  - LocationID
  - Euclidean distance


10
5
15
1
|10 - 1|
|5 - 1|
|15 - 1|
50
|10 - 50|
|5 - 50|
|15 - 50|
reshape
9
4
14
40
45
35
stats
{
"feature_max": 45,
"feature_std": 16.028620235898867,
"feature_min": 4,
"feature_mean": 24.5
}

Features 2: Attributes

	key	count	nuniq	items
0	Вид одежды	364403	3	{Мужская одежда, Аксессуары, Женская одежда}
1	Предмет одежды	338387	15	{Брюки, Топы и футболки, Пиджаки и костюмы, Др...
2	Размер	323455	42	{40–42 (XS), > 34, 42, 50–52 (XL), > 38, 46–48...

- Regularized Jaccard of keys and key=value pairs

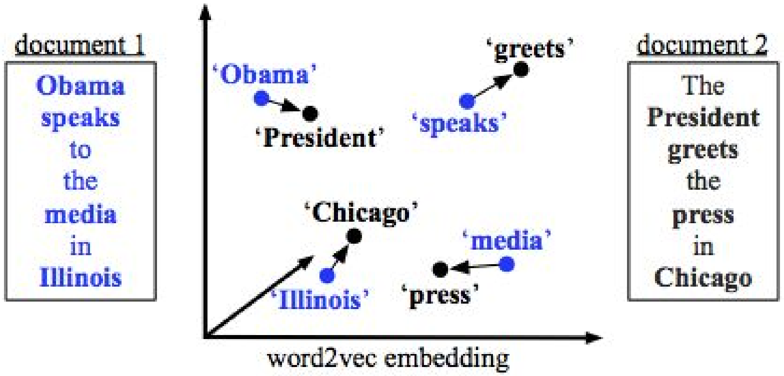
$$J_\lambda(X, Y) = \frac{|X \cap Y|}{|X \cup Y| + \lambda}$$

- Number of fields both ads didn't fill
- TF-IDF on key=value pairs
  - dot product in TF-IDF space (`norm=None`) was better than cosine
- Cosine in SVD of TF-IDF

Features 3: Text

- Jaccard & Cosine on digits only and on English tokens only
- Russian chars in English words
  - E.g. “o” in “iphone” is Cyrillic
- Cosine in TF, TF-IDF, BM25, cosine of SVD of them
- Common tokens & differences:
  - Text1: “продам iphone”, Text2: “продам айфон”
  - Common: {продам}, Difference: {iphone, айфон}
  - Cosine in TF (binary), SVD of it
- Word2Vec & GloVe
  - Cosine and manhattan b/w average title vectors
  - Stats of pairwise cosines between all tokens excluding the same ones
  - Tokens from title, description, title + description, nouns only

Features 3 cont’d: Word’s Mover Distance



- “True” WMD is complex and slow
- “Poor Man’s” WMD is faster:
  - WMD(A, B): For each term in doc A take distance to closest term in doc B, sum over them
  - WMD_sym(A, B) = WMD(A, B) + WMD(B, A)

figure from https://github.com/mkusner/wmd

Features 3 cont'd: Misspellings

- Idea: same author can make same types of mistakes
  - No space after dot/comma ("продам айфон.дешево")
  - Morphological errors
  - And others
- Represent ads as "Bag of Misspellings"
- Use Regularized Jaccard and Cosine
- Misspellings extracted with languagetool.org

Features 4: Images

- Stuff that everybody used
  - Image hashes from `imagehash` library and forums
  - Chi2 & Bhattacharyya on histograms (with openIMAJ)
  - SIFT keypoints + matching (with openIMAJ)
  - Structural Similarity (computed with pyssim)

- Perceptive hashes computed with imagemagick
  - hashes computed on each channel separately and on the mean channel
- Image moments: Centroids (“Ellipses” in imagemagick)
  - Centroids = centers of masses of each channel
  - Distances between image centroids in each channel
- Image moment invariants (imagemagick)
  - 7 moments, invariant to translation, scale and rotation
  - Put all 7 invariants in a vector, compute cosine and distance

Features 5: GEO

- Reverse (lat, lon) code to location

(53.87, 27.66) → OpenStreetMap →

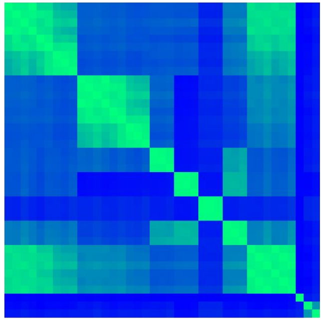

```
"address": {
  "city": "Минск",
  "house_number": "82",
  "country": "Беларусь",
  "suburb": "Северный Посёлок",
  "city_district": "Заводской район",
  "postcode": "220137",
  "country_code": "by",
  "road": "Алтайский переулок"
}
```


→ (region, city, zip)

- Features like same_region, same_city, same_zip
- |zip1 - zip2|

Feature Selection

- Correlation
  - A lot of features. Many correlated ones
  - Find feature groups of 0.90 correlation
  - Keep only one of the features
- XGBoost Feature Importance
  - https://github.com/Far0n/xgbfi
  - Run xgb on a sample with 100 trees
  - Use xgbfi to extract most important features
- Combined
  - In a correlated group, choose the most important feature using xgbfi output



Most Important Features

1	Interaction	Gain
2	imagemagick_abs_diff_ellipse_green_amin	9610555.82
3	svm_all_text_common	8216380.23
4	svm_title_both	4946313.60
5	kp_matched_mean	2583469.44
6	all_text_1_all_text_2_token_set_ratio	2158867.14
7	category	1884346.64
8	price_diff	1603385.08
9	svm_title_diff	1284061.05
10	imagemagick_no_exact_matches	907893.61
11	w2v_title_1_title_2_euclidean_amin	822576.55
12	attrs_pairs_manhattan_tfidf_svd	762041.06
13	imagemagick_phash_all_pairs_manhattan_amin	485939.15
14	attrs_values_dot_tfidf	456179.96
15	title_1_title_2_UWRatio	406206.79
16	imagemagick_abs_diff_imstat_green_skewness_amin	387381.94
17	w2v_title_1_title_2_manhattan_amin	381082.32
18	svd_all_text_3	371899.74
19	attrs_values_jaccard_reg	368998.79
20	imagemagick_abs_diff_imstat_red_kurtosis_amin	328200.49
21	text_all_bm25_dot	319062.96
22	text_all_tfidf_cosine	278849.15
23	w2v_all_text_1_all_text_2_euclid_wmd_mean	232960.93
24	attrs_values_num_match	231059.75
25	imagemagick_abs_diff_imstat_overall_kurtosis_amin	212947.41
26	attrs_pairs_num_match	211269.95

SVM fit on common & diff tokens

Models & Ensembling

- Parameter tuning
  - Random search
- My best model: 0.939 public LB
  - XGB with depth=8 and 2.5k trees
  - Trained a few days
- Ensembling:
  - Sample group of features
  - Randomly choose the parameters
  - Build ETs and XGBs
  - Stack with Log Reg (L2 regularization with low C)
- Our final model:
  - Neural network on ETs and XGBs outputs + some selected 1st level features

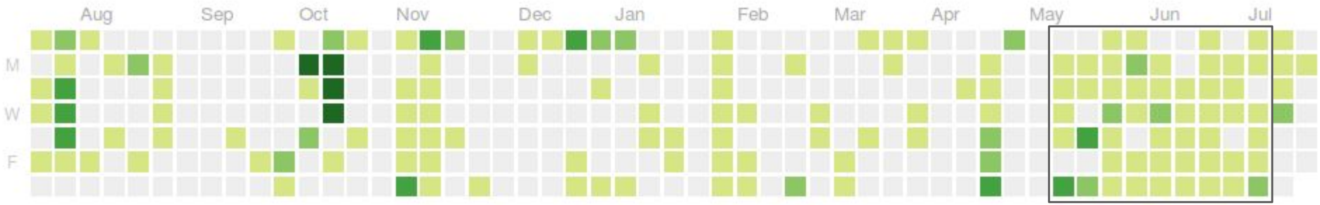
Lessons Learned

- It’s important to get CV right
  - My scheme: shuffle 3 fold (leaky)
  - Couldn’t use some nice features because of it
  - CV score of the ensemble was too good
  - Result: CV via LB
  - The right one: by connected components
- A lot of features is not always good
  - Computed too many features
  - Had hard time managing to use them all
  - Had to start stacking early

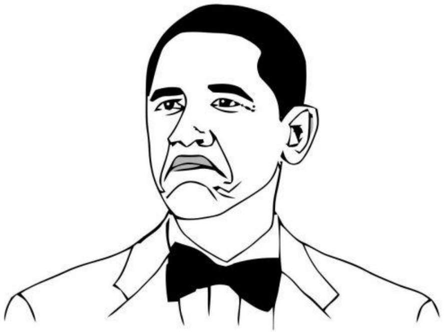
	itemID_1	itemID_2
0	1	4112648
1	3	1991275
2	4	1223296
3	7	1058851
4	8	2161930
5	9	694103
6	12	5637025
7	12	5279740

That’s a graph!

#	Δrank	Team Name ‡ model uploaded * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	—	Devil Team *	0.95829	162	Mon, 11 Jul 2016 23:09:03 (-0.7h)
2	—	TheQuants *	0.95294	197	Mon, 11 Jul 2016 19:53:15 (-46.7h)
3	—	ADAD *	0.94971	226	Mon, 11 Jul 2016 23:57:54
4	—	8 + 9 = 11	0.94694	193	Mon, 11 Jul 2016 13:01:32 (-6.3h)
5	—	**ololobhi** • **Abhishek** • **ololo**	**0.94587**	**133**	**Mon, 11 Jul 2016 22:18:01**
6	—	otivA	0.94560	117	Mon, 11 Jul 2016 13:09:48 (-0.1h)
7	—	Native Russian Speakers :P	0.94449	43	Mon, 11 Jul 2016 22:49:32 (-1.3h)
8	↑1	frist	0.94438	158	Mon, 11 Jul 2016 21:28:56 (-1h)
9	↓1	DataMinders	0.94411	244	Mon, 11 Jul 2016 17:51:57
10	↑1	Pavel Blinov	0.94272	91	Mon, 11 Jul 2016 20:02:24 (-0.1h)



https://github.com/alexeygrigorev/avito-duplicates-kaggle



Questions?