Definition

Project Overview

Recommendation engines have been used to recommend a variety of things from movies, music, articles, products or even a company's next potential employee.

In general there is typically two types of recommendation engines:

Content Based (CB)

Content based algorithms usually work best when recommending similar or related items of the same type or category where the properties or context of each item can easily be determined.

The advantage of this approach is no user data is really required to generate recommendations, however if this approach is taken then the disadvantage is that it doesn't really help the user discover new products that they may be interested in based on their preferences or previous actions such as ratings or product browsing history.

Collaborative Filtering (CF)

Collaborative filtering requires some sort of user input or action to be taken, then based off a collection of the user's past behaviour it can then produce recommendations that are best suited for that user.

Collaborative filtering algorithms can be further broken down into several groups the two main ones are:

- **User-User Collaborative Filtering:** whereas if Person A purchases items 1, 2, 3 and Person B purchase items 2, 3, 4 both Person A and B share similarities (they both purchased items 2 and 3) therefore it is likely the Person A might be interested in item 4 and Person B may be interested in item 1.

- **Item-Item Collaborative Filtering**: instead of matching similar users item-item collaborative filtering matches products that a user has purchased or rated to other similar items. Item-item collaborative filtering was invented and used by Amazon.com in 1998[1]

[1] Collaborative recommendations using item-to-item similarity mappings - https://www.google.com/patents/US6266649

The disadvantage of collaborative filtering is they can be relatively heavy on computational power they also require a large amount of user data, an obvious advantage though is recommendations can be more personalised on a per user bases.

It is also worth mentioning that a hybrid approach can also be taken where CB and CF are both used in conjunction with each other to produce a hybrid recommendation system. Once again these systems require large amounts of data and computational power to run the various recommendation systems.

For our particular use case and the dataset we have to work with, we will be creating a CB recommender. This is the best option given we don't really have any user data and we are only concerned with recommending items in the same category related to a particular product.

We will explore how we can apply machine learning techniques to build our CB recommender based solely off of the product's content given to us, in this case we dealing with 65 products under the Andriod TV Box category. We will build *Item Profiles* by extracting the item features. In our case we are dealing with products so features can consist of product attributes such as RAM size, CPU speed etc. So a valid starting point to extract these features would be the product's specification attributes. We will also examine how using other feature selections such as the product's meta-keywords or overview compare in result.

Problem Statement

"An efficient way to automate the selection of related items for a product that outperforms the manual approach by measurement of similarity"

Background

The problem we wish to solve is for an ecommerce store selling electronics we have selected a single category as a starting point for this project, "Android TV Boxes".

The current manual approach requires the user to input the details of a product and based on its specifications select up to three products to be listed as a related item. In the case of the Tv boxes this might mean having two products that can both be used for viewing on a 4K resolution Tv screen but one may have 2GB ram and the other 1GB ram amongst other specifications.

The main pain points of this process are:

1. the user entering the information may be unfamiliar with the product category;
2. it is time consuming having to manually browse and search products with similar specification

As a result this step is sometimes skipped or less related products are likely to be selected.

Metrics

We evaluate the performance of each model by measuring the training time in seconds and measure the scoring of our similarities between products using cosine similarity function which is readily available from scikit learn.[2]

Cosine similarity measures the distance between 2 points by calculating the angle from the origin. The smaller the angle the more similar two items are likely to be, larger the angle less similar they are likely to be. The reason to use cosine similarity is we are only interested on how each text relates to each other without the effect of how long the text is. In other words we are not concerned with the magnitude of the text just the directional similarity.

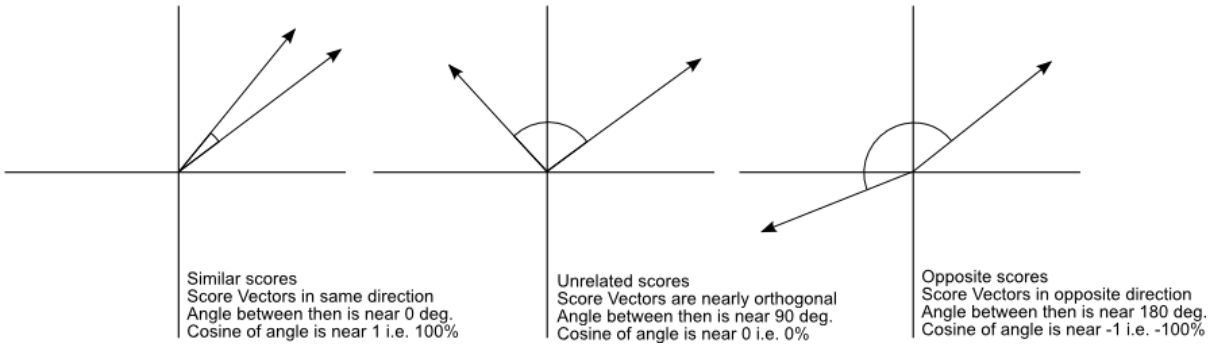


Figure 1. Cosine Similarity - scores according to angles between two points.

We also measure the sparsity[3] of each model which is calculated by taking the number of zero-valued elements in a sparse matrix and dividing it by the total number of elements. Selecting a model with the least amount of sparsity should result in more similar items being selected.

Analysis

Data a visual exploration

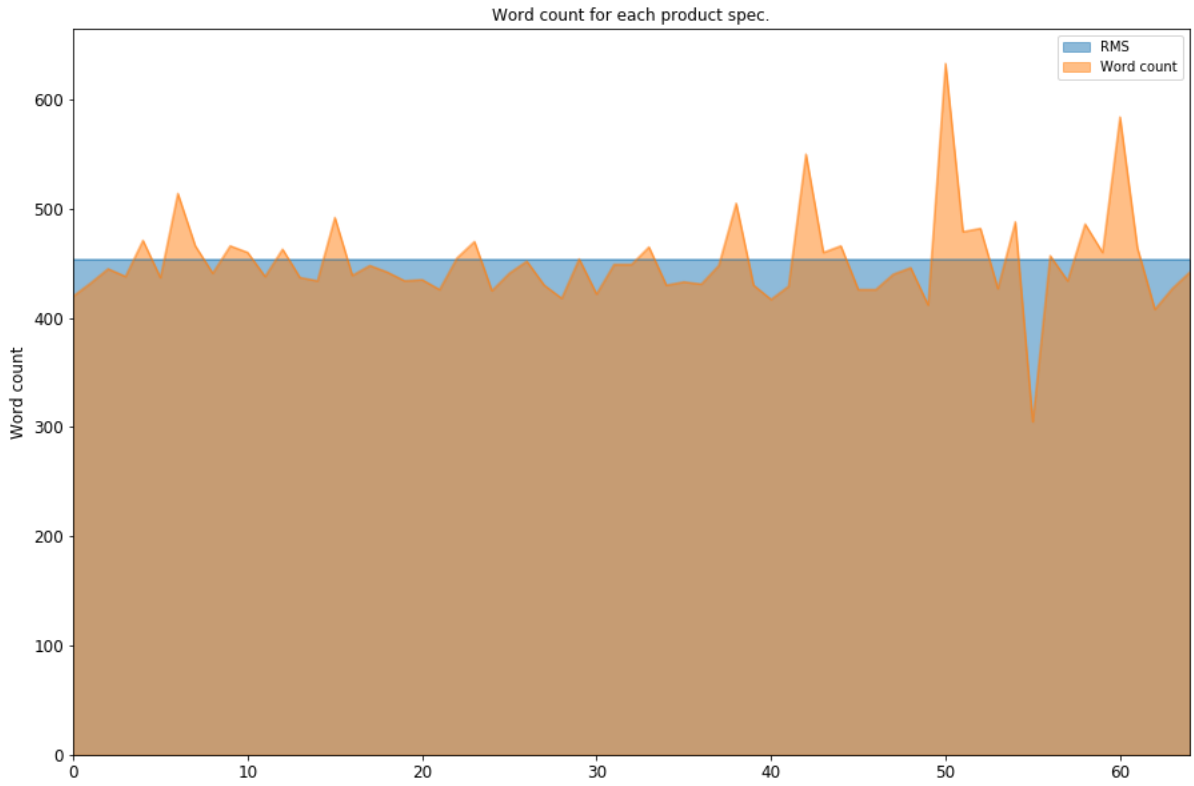
The dataset we are working with has been provided by the client in a *products.json* file that consists of a total of 65 products under the Andriod TV Box category. Although a lot of data has been provided the core of the content that we will be working with to serve our purpose is the product's:

[2] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
[3] https://en.wikipedia.org/wiki/Sparse_matrix

- full and short name
- overview/description
- related products (these have been manually selected)
- specification
- meta keywords and description

The product overview and specification are formated in HTML which had to be parsed. For the overview this was simply just a matter of removing the HTML tags to have a readable plain text version, with regards to the specification where most of the features have been extracted further parsing was required. Luckily there is a python library Beautiful Soup[4] which made extracting the information needed rather simple.

Since most of the features came from the each product's specification one simple way to see if each product's written specification is similar at least in terms of wording is to check the word count. Looking **Figure 2** we can see that the majority of the word counts for each product specification is close to the overall root mean square (RMS), but in some instances we do have some outliers that are either way above or below the RMS count. This indicates that some products may have less or more attributes.



RMS word count is 453 | Average word count is 451 | Maximum word count is 633| Minimum word count is 305

Figure 2. Word count for each product's specification (after parsing HTML tags) compared to RMS.

Let's further examine the specifications for the product that had the highest and lowest word count, so we can contrast them against each other.

[4] https://www.crummy.com/software/BeautifulSoup/

Product specification for product with the highest and lowest word count.		
Product ID	20352	20656
Word count	633	305
Spec.	**Manufacturer Specifications** **General** ● OS Version: Android 5.1 ● CPU: Amlogic S905 Quad Core 64Bit ● GPU: Penta-Core ARM Mali 450 ● Processor Speed (max): 2.0GHz ● RAM: 1GB ● Internal Memory: 8GB ● External Memory: Support up to 32GB ● Display Resolution: 4K (2880 x 2160P) ● Supported Video Resolution: 4K x 2K ● Support H.265 ● Wi-Fi: 802.11 b/g/n ● Google Play ● DLNA ● Miracast ● Airplay ● Kodi / XBMC V15.2 ● DVB: DVB-T2, DVB-S2, ISDB-T, DTMB, ATSC (OPTION) **DVB-S2 (OPTION)** ● DVB-S2 System Standard: ETS 302 307 ● Symbol Rates: QPSK, 8PSK ● Forward Error Correction Rate: 1/4, 1/3, 2/5, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10 ● Carrier Frequency Acquisition Range: +/-5MHz for symbol rates above 3 Msps and +/-3MHz for the remaining symbol rates ● Roll-off factors for pulse Shaping: 0.2, 0.25, 0.35 ● Symbol Rates: 1 to 55 Msps ● Code Rates: 1/2, 2/3, 3/4, 5/6, 7/8 ● Input Signal Level: -84 dBm to 0 dBm (average power) ● ANT Connector: IEC TYPE F **DVB-T2 (OPTION)** ● DVB-T2 System Standard: ETSIEN 302 755 ● Support MPLP ● Demodulation: QPSK, 16QAM, 64QAM or 256QAM ● Forward Error Correction rate:	**Manufacturer Specifications** **General** ● OS Version: Android 5.1 ● CPU: S905 Quad Core Cortex-A53 ● GPU: Mali-450MP ● Processor Speed (max): 2.0GHz ● RAM: 2GB ● Internal Memory: 32GB ● External Memory: Support up to 64GB ● Display Resolution: 4K x 2K ● Supported Video Resolution: 4K x 2K ● WiFi: 802.11 b/g/n ● Bluetooth 4.0 ● Google Play ● Kodi 16.0 / XBMC **Ports** ● HDMI ● Micro SD Card Slot: up to 64GB ● 3 x USB Port ● MRJ45 port ● Audio R ● Audio L ● Video ● SPDIF ● DC IN **Languages** ● Bahasa Indonesia, Bahasa Melayu, German, English, Spanish, Filipino, French, Italian, Magyar, Dutch, Portuguese (Brasil), Portuguese (Portugal), Vietnamese, Turkish, Greek, Russian, Hebrew, Arab, Thai, Korean, Chinese (Simp), Chinese(Trad), Japanese **Media Formats** ● Video: AVI, RM, RMVB, TS, MKV, VOB, MOV, ISO, WMV, ASF, FLV, DAT, MPG, MPEG ● Audio: MP3, OGG, WMA, AAC, WAV, AC3, DDP, TrueHD, DTS, HD, FLAC, APE

	1/2,2/3,3/4,3/5,5/6 ● Guard Interval: 1/4, 19/256, 1/8, 19/128, 1/16, 1/32, 1/128 ● Band: 6/7/8MHz ● Input frange Range: 54 to 860 MHz ● Input Impedance: 75 Ohm ● Input Signal Level: -84 to 0 dBm (average power) ● ANT Connector: IEC TYPE, female **Ports** ● DVB-T2 ● DVB-T2 ● CVBS/L/R ● LAN ● HDMI ● OPTICAL ● DC IN ● Micro SD card slot ● 4x USB **Languages** ● Bahasa Indonesia, Bahasa Melayu, German, English, Spanish, Filipino, French, Italian, Magyar, Dutch, Portuguese (Brasil), Portuguese (Portugal), Vietnamese, Turkish, Greek, Russian, Hebrew, Arab, Thai, Korean, Chinese (simp), Chinese (trad), Japanese **Media Formats** ● Video: AVI, RM, RMVB, TS, VOB, MKV, MOV, ISO, WMV, ASF, FLV, DAT, MPG, MPEG ● Audio: MP3, WMA, AAC, WAV, OGG, DDP, TrueHD, HD, FLAC, APE(AC3 and DTS OPTION) ● Graphic: JPEG, BMP, GIF, PNG, TIFF **Dimensions** ● Main Product Dimensions: 108 x 130 x 30 mm (L x W x D) ● Main Product Weight: 185g ● Weight/dimension is for the main item of this boxed product, to help you compare product sizes before buying: please do not base your shipping calculations on this price - shipping prices depend on your cart contents, shipping destination, and shipping method: please use the checkout to select options and preview shipping price for your total order. **Product Notes**	● Graphic: JPEG, PNG, BMP, GIF, TIFF ● eBook: PDF, TXT **Dimensions** ● Main Product Dimensions: 162x131 x24 mm (L x W x D) ● Main Product Weight: 370g ● Weight/dimension is for the main item of this boxed product, to help you compare product sizes before buying: please do not base your shipping calculations on this price shipping prices depend on your cart contents, shipping destination, and shipping method: please use the checkout to select options and preview shipping price for your total order. **Product Notes** ● The Android OS version on this device cannot be upgraded or flashed and any attempts to modify the default OS will void the warranty. As a wholesaler, we provide no software support, advice, or training regarding the Android operating system and software. **Package Contents** ● TV Box ● Remote Control ● Power Adapter ● HDMI Cable ● User Manual **Enjoy the following benefits:** ● Orders processed and shipped within 1 Working Day ● 12 month warranty ● Inhouse QC ● Member discounts ● Award winning customer support ● Quantity order discounts ● Worldwide Shipping ● Certification: CE, FCC, RoHS **Foreign Language Keywords**

	• The Android OS version on this device cannot be upgraded or flashed and any attempts to modify the default OS will void the warranty. As a wholesaler, we provide no software support, advice, or training regarding the Android operating system and software. **Package Contents** • TV Box • Remote Control • Power Adapter • HDMI Cable • User Manual **Enjoy the following benefits:** • Orders processed and shipped within 1 Working Day • 12 month warranty • In-house QC • Member discounts • Award winning customer support • Quantity order discounts • Worldwide Shipping • Certification: CE, FCC, RoHS **Foreign Language Keywords** Arabic: - مربع التلفزيون الروبوت 64 بت رباعية النوى - Chinese Simplified: - 四核安卓的 64 位电视盒 - Czech: - Čtyřjádrový 64bitový Android TV Box - Danish: - Quad Core Android 64 Bit TV boks - Dutch: - Quad Core Android 64 Bit TV Box - French: - Quad Core 64-Bit Android TV Box - German: - Quad-Core 64-Bit-Android TV-Box - Hebrew: - ביט 64 ליבות אנדרואיד טלוויזיה תיבת - Hindi: - Quad कोर एंड्रॉयड 64 बिट टी वी बॉक्स - Italian: - Quad-Core a 64 Bit Android TV Box - Japanese: - クアッドコア 64 ビットのアンドロイド テレビ ボックス - Korean: - 쿼드 코어 64 비트 안 드 로이드 TV 박스 - Malay: - Peti TV teras Quad Android 64 Bit - Norwegian Bokmål: - Firekjerners Android 64 Bit TV boks - Romanian: - Quad Core Android 64 Bit TV Box - Russian: - Четырехъядерных андроид 64 Bit TV Box - Spanish: - Quad Core 64bits Android TV Box - Thai: - กล่องทีวีหุ่นยนต์ 64 บิตหลักสี่ - Turkish: - Dört çekirdekli Android 64 Bit TV kutusu - Vietnamese: - Lõi tứ 64-Bit Android TV Box	

Table 1. Comparing product specifications for the highest and lowest word count.

Quick browse over the table we can see that there is a lot of information within the product specifications that although useful to the user it doesn't represent the characteristics of the product itself. Clear examples would be "Foreign Language Keywords", "Enjoy the following benefits","Package Content" etc. information that better describes the attributes of the product are "General" specs. "Ports" available on the devices and since the product is used mainly for viewing media, supported "Media Formats".

Following this pattern we check if all products indeed contain this information within their specifications. Taking a closer look at the product specification in HTML we can see that each listed items parent tag (<ul>) contains an ID, as represented in the below example excerpt.

```
<strong>General</strong>
<ul id='general'>
  <li>OS Version: Android 5.1</li>
  <li>CPU: Rock Chip 3229 Quad-Core</li>
  ...
</ul>
```

With this knowledge we can simply loop over each product specification and check if the id tags exist for **General, Ports** and **Media Format**

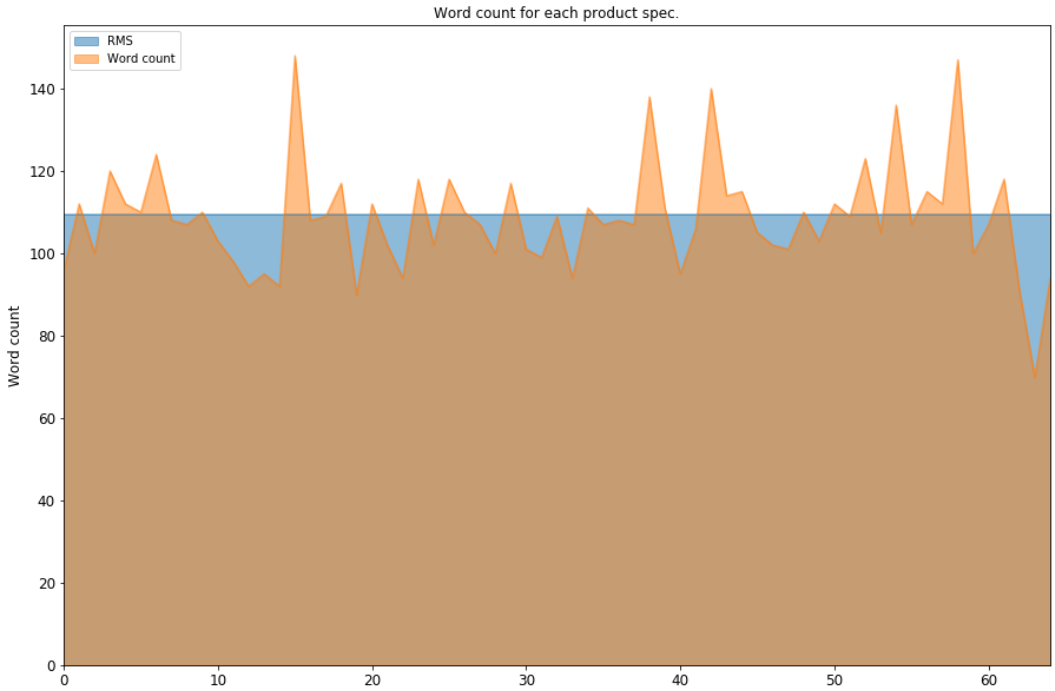
As represented by the pseudo code below we run our loop and confirm that each product does indeed have these headings within their specification.

```
count_missing_ids = 0
for each product:
    if not id_exists(id):
        count_missing_ids += 1

If count_missing_ids > 0:
    print “id is missing for count_missing_ids product(s)”

# Returns
# All products have general spec.
# All products have Ports spec.
# All products have Media_Formats spec.
```

After extracting the data if we re-examine **Figure 3.** we can see that each product specification still varies from product to product but some do share close to the same word count showing similarities between the specifications.



RMS word count is 109 | Average word count is 108 | Maximum word count is 148 | Minimum word count is 70

Figure 3. Word count for each product's refined specification (after parsing HTML tags) compared to RMS.

Algorithms and techniques

Given we are interested in determining the similarities between products based on the written content provided, the best approach will be to use the *term frequency–inverse document frequency (TF-IDF)* algorithm which works very well at extracting features from text.

Very briefly TF-IDF work as follows:

Given we have a keyword ***w*** and a document ***d***:
TF (term frequency): Measures, how often a term appears.

- TF(***w, d)*** = term frequency of keyword ***w*** in document ***d***

IDF (inverse document frequency): Aims to reduce the weight of terms that appear in all documents.

- IDF(***w***) = $log\frac{N}{n(w)}$
    - ***N*** = number of all recommended documents
    - ***n(w)*** = number of documents from ***N*** in which keyword ***w*** appears

Finally TF-IDF(***w, d***) is calculated as TF(***w, d***) * IDF(***w***)

Usually we would try to gain a bag-of-words which in simple terms counts how often a word appears in a each text, then the bag-of-words would be taken and transformed by *tf-idf.*

However *scikit-learn* provides a *TfidfVectorizer*[5] which takes in the text data and does both the bag-of-words feature extraction and the *tf–idf* transformation.

Benchmark model

For the benchmark model we will only remove english stop words, and set the analyzer to words, the *TfidfVectorizer* was fitted with each product's **Meta Keywords**.

Abbreviated list of stop words that will be removed:

```
['above', 'in', 'inc', 'third', 'back', 'now', 'per', 'while', 'yours', 'about',
'this', 'everything', 'nor', 'thick', 'last', 'least', 'formerly', 'whereby',
'mine', 'move', 'never', 'beside', 'must', 'herself', 'find', 'whose', 'would',
'otherwise', 'hereupon', 'might', 'because', 'which', 'eight', 'out', 'thus',
'nobody', 'through', 'seems', 'same', 'becoming', 'they', 'became', 'anywhere',
'before', 'anyhow', 'fifty', 'no', 'if', 'among', 'or', 'see', 'sincere',
'these', 'give', 'thereby', 'de', 'fifteen', 'often', 'whence', 'but', 're',
'such', 'couldnt', 'etc', 'yet', 'me', 'after', 'perhaps', 'yourself', 'upon',
'very', 'other', 'how', 'done', 'those', 'latter', 'on', 'myself', 'first',
'many', 'into', 'again', ...]
```

List of feature words (bag-of-words) that will be weighted by tf-idf, total size of vocabulary is 115 terms:

```
['10', '1080p', '3d', '4k', 'a200', 'accessory', 'airplay', 'am3',
'amlogic', 'andriod', 'android', 'band', 'bar', 'bb2', 'best', 'bluetooth',
'box', 'boxes', 'budget', 'center', 'cheap', 'computer', 'coowell', 'core',
'dual', 'dvb', 'entertainment', 'f7', 'fi', 'google', 'h96', 'hd', 'hdmi',
'home', 'hub', 'internet', 'k1', 'k3', 'kb2', 'km5', 'km8', 'kodi', 'ks1',
'm8spro', 'mecool', 'medi', 'media', 'meegopad', 'micro', 'mini', 'minix',
'multimedia', 'mx', 'mx3', 'mx3tv', 'mx9', 'mxiii', 'mxq', 'neo', 'octa',
'ott', 'pc', 'pip', 'player', 'plus', 'premium', 'pro', 'quad', 'r99',
'recording', 's2', 's905', 's912', 'scishion', 'smart', 'sound', 'soundbar',
'star', 'stick', 'sunvell', 't09', 't2', 't95max', 't95n', 't95p', 't95u',
't95v', 't95z', 'television', 'tv', 'u1', 'u9', 'ugoos', 'uhd', 'ultra',
'ut6', 'v88', 'v99', 'wholesale', 'wi', 'windows', 'wireless', 'x7', 'x8',
'x9', 'x92', 'x95', 'x96', 'x98', 'x9s', 'xbmc', 'yoka', 'yokatv', 'z4',
'zidoo']
```

[5] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Top 10 terms by weighted score.		
Index	**Term**	**Weight**
89	tv	30.182777
16	box	27.390094
10	android	17.244235
3	4k	6.887078
46	media	5.111881
63	player	4.906306
74	smart	4.584562
14	best	4.272599
41	kodi	3.911404
20	cheap	3.507161

Sparse matrix shape: (65, 115) | Nonzero count: 605 out of 7475 | Sparsity: 91.91%

Table 2. Top 10 weighted terms.

Sparsity (ratio of terms that received zero as the weight) can be reduced by removing terms that occur less frequent or too often. The higher the sparsity the more challenging it becomes for our model to measure the similarity between products.

We can also see that our benchmark model selects more related items than the manual approach when using product ID 20673 as an example.

Manually selected items vs. selected by model (Parent product: 20673)						
	Top 3 products selected by model			**Top 3 Manually selected products**		
Product ID	20729	20781	20654	19830	20242	20654
Similarity score	0.953	0.913	0.895	0.672	0.789	0.805
Text parsed in (seconds):	0.24680					
Engine trained in (seconds):	0.00674					

Table 3. Similar items selected by model vs. manual selection.

Methodology

Data Preprocessing

Since the features were extracted from the product specification which is in *HTML* it had to be parsed by 1) extracting only the contented we needed to work with and 2) removing all *HTML* tags. Using an example product **Table 3.** shows the process of the final parsed text starting from the original *HTML* data.

Extracting and parsing the *HTML* text from the product's specification.		
Original specification	**Extracted content**	**Final parsed text**
 Manufacturer Specifications <ul id="Manufacturer_Specifications"> General <ul id="general"> OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K ...	<ul id="general"> OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play ...	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play Kodi 16.1 3D Movie Support HDMI RJ45 port AV out Micro SD Card Slot: up to 16GB 4xUSB Port SPDIF DC IN Video: MKV, WMV/VC-1 SP/MP/AP, MPG, MPEG, DAT, AVI, MOV, ISO, MP4, RM, H.264, H.265, RealVideo 8/9/10, VP8/9, up to 2160P Audio: MP3, AAC, WMA, RM, FLAC, OGG Graphic: JPG, JPEG, MJPEG, PNG

Table 4. The process of parsing the HTML text for a product's specification. HTML has been abbreviated for readability.

Implementation

Process

The steps taken to determine the best model that we can further tune were:

Step 1) Fit the *tf-idf vectorizer* with text. The text was selected from four different areas and fitted separately. These areas were the product's overview, meta keywords specification and finally the refined specification. The only parameters set were the analyzer to words and english stop-words were also removed.

Step 2) As explained in the section **Metrics** we calculate the similarity score for each product, as well as the sparsity.

Step 4) Select the model with the least sparsity.

Step 5) Refine model by tweaking its parameters.

Selecting the best model

Below are the results used to determine the best model that we can further improve on.

Model 1. Using product's meta keywords (**benchmark model**):

Model performance using meta keywords	
Text parsed in (seconds):	0.00615
Engine trained in (seconds):	0.00234
Vocabulary size (word count):	115
Sparsity (%):	91.91

Table 5. Model performance using meta keywords to fit TfidfVectorizer.

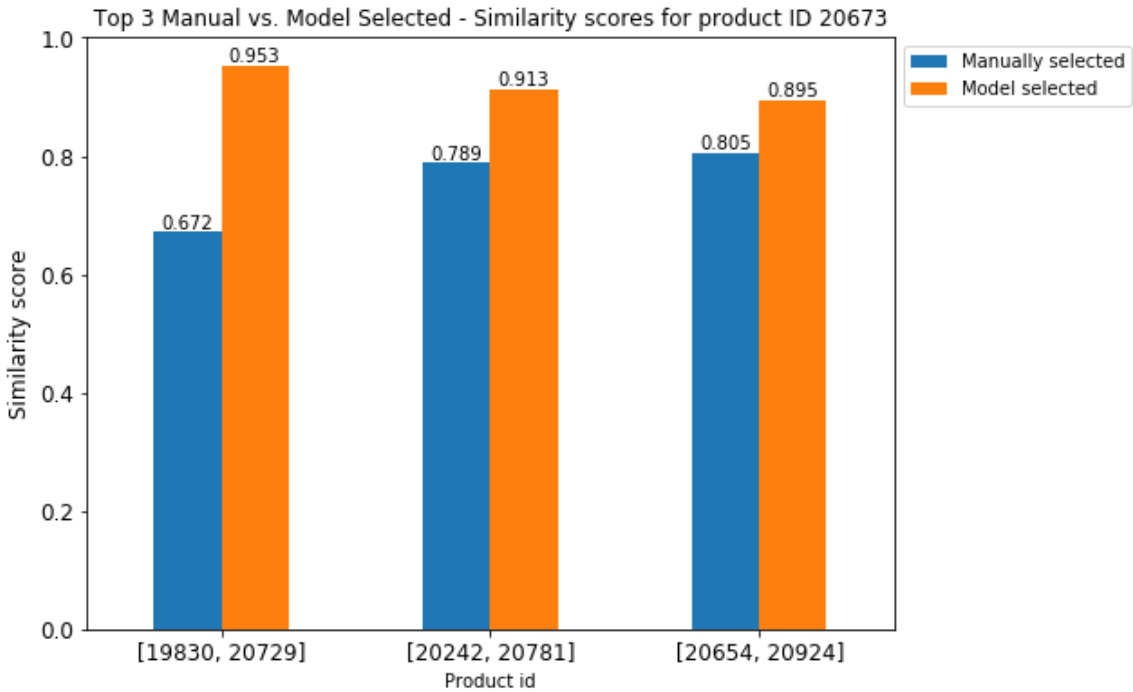


Figure 4. Manual vs. Model selection of related items, using Meta Keywords.

Model 2. Using product's overview

Model performance using overview	
Text parsed in (seconds):	0.03764
Engine trained in (seconds):	0.01893
Vocabulary size (word count):	1651
Sparsity (%):	90.84

Table 6. Model performance using overview to fit TfidfVectorizer.

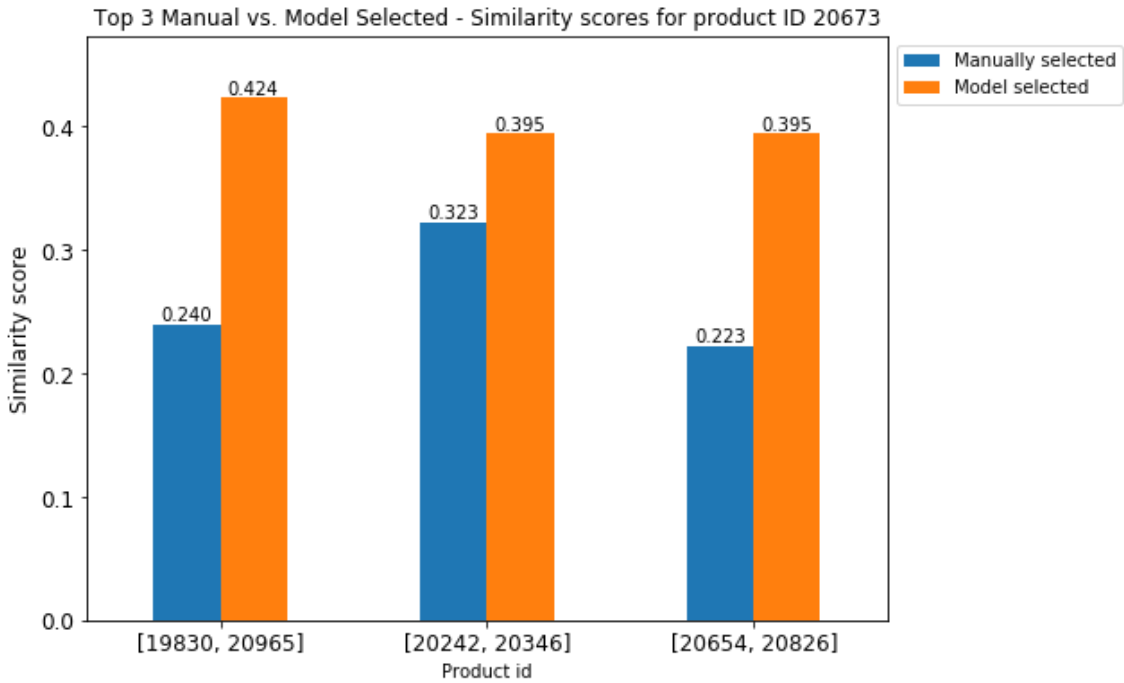


Figure 5. Manual vs. Model selection of related items, using Overview.

Model 3. Using product's specification

Model performance using specification	
Text parsed in (seconds):	0.13621
Engine trained in (seconds):	0.01992
Vocabulary size (word count):	956
Sparsity (%):	74.42%

Table 7. Model performance using specification to fit TfidfVectorizer.

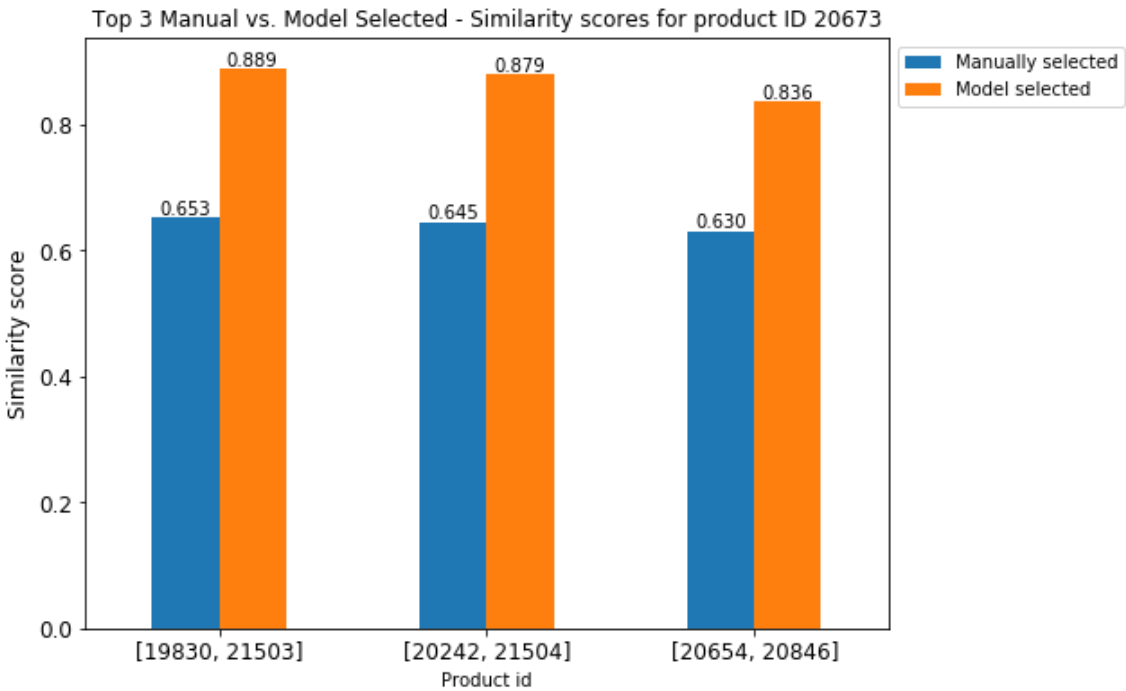


Figure 6. Manual vs. Model selection of related items, using Specification.

Model 4. Using product's refined specification

Model performance using refined specification	
Text parsed in (seconds):	0.24680
Engine trained in (seconds):	0.00674
Vocabulary size (word count):	359
Sparsity (%):	74.27%

Table 8. Model performance using refined specification to fit TfidfVectorizer.

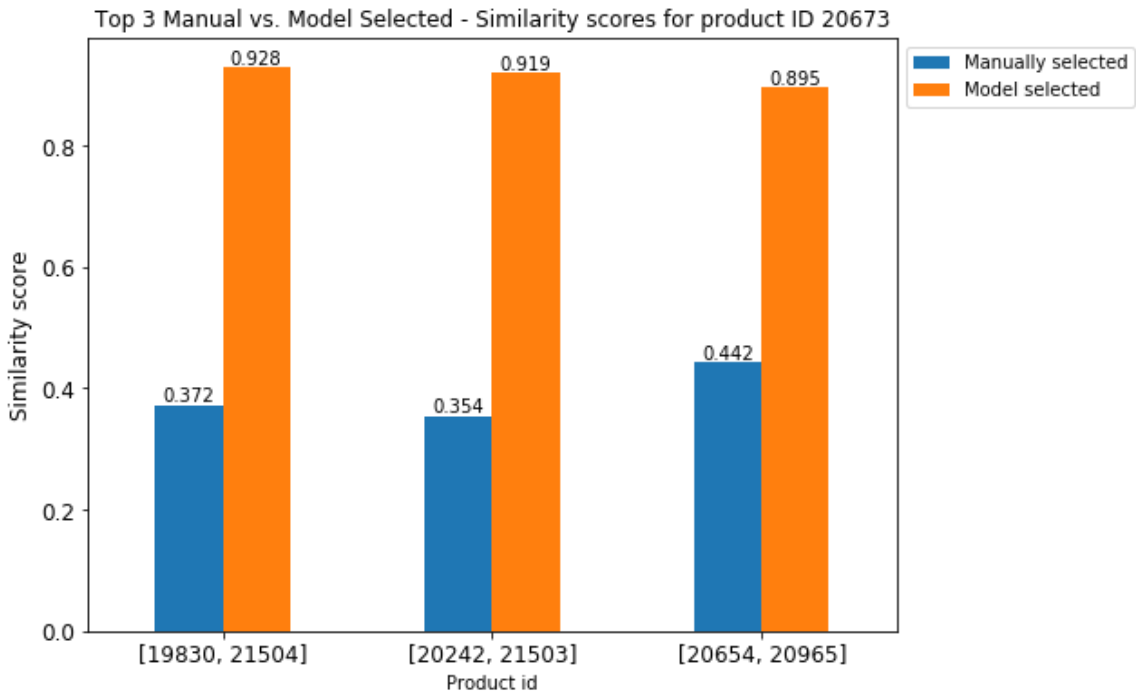


Figure 7. Manual vs. Model selection of related items, using Refined Specification.

In conclusion using product ID 20673 as an example we can see all our models managed to selected items the were more similar to that of the manual approach. **Model 3** and **Model 4** produced the lowest sparsity where **Model 4** was only slightly better by a small margin, however it did produce items that were much more similar and we also see an increase in similarity scores which could suggest **Model 4** is able to calculate similarities much better.

Refinement

Although all our models were selecting more relevant items than the manual approach when using product ID 20673 as an example, we can further improve our results by tweaking the parameters of the *TfidfVectorizer.* **Model 4** above was selected to further improve upon.

As an initial step to refine the Model we can remove any words that do not appear at least in two product refined specifications, if a word only appears once in a product's refined specification that feature is completely unique to that product and holds no weight when trying to determine other related items. This reduced our vocabulary from 359 terms to 240 and reduced sparsity from 74.27% to 62.28% which is 29.63% less sparsity than our benchmark model too.

Again in **Figure 8** we us product ID 20673 as an example to see how our new model compares to the original **Model 4**.

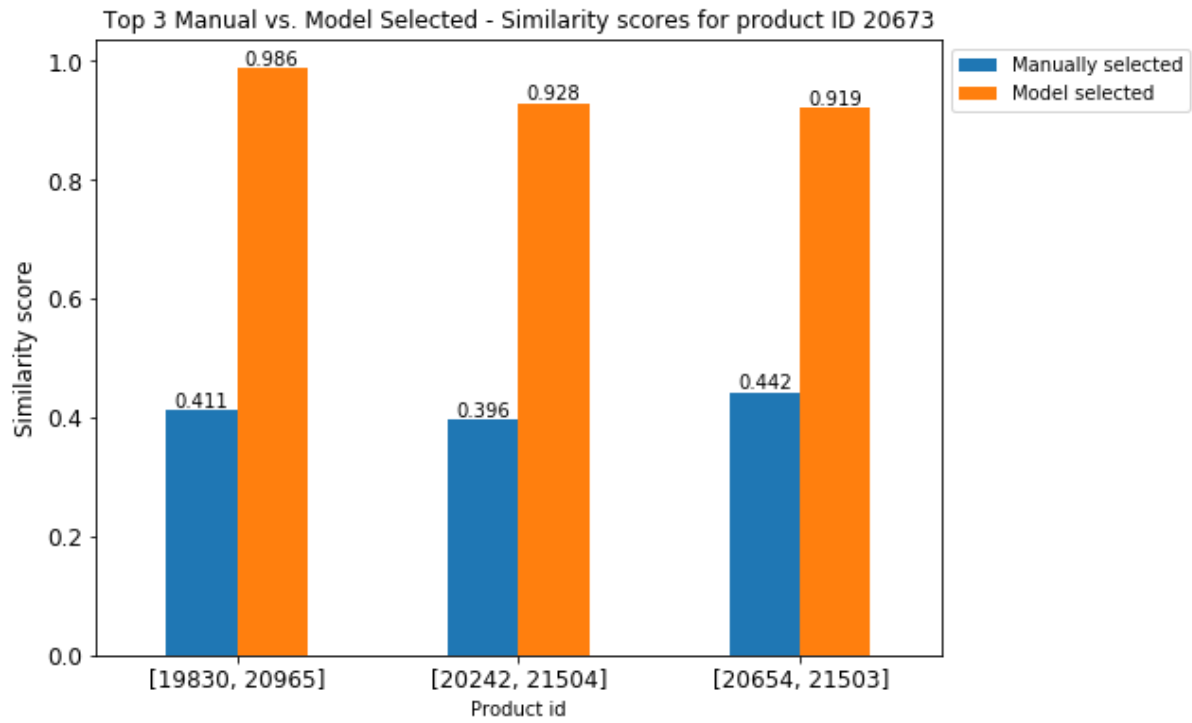


Figure 8. Manual vs. Model selection of related items, using Refined Specification.

If we look at **Figure 7** we can see product ID 20965 with similarity score 0.895 has pushed to first place with a new similarity score of 0.986 in **Figure 8**. In **Table 9** we do a quick sanity check by comparing the refined specifications of product ID 20965 and 21504 to that of the parent product's (20673) refined specification.

Refined specification comparison for product ID 20673.			
Product ID	**20673**	20965	21504
Product name	**V88 Android TV Box**	SCISHION V88 Plus Android TV Box	4K Android TV Box T96R
	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play Kodi 16.1 3D Movie Support ...	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 2GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play Kodi 16.1 3D Movie Support E home media center ...	OS Version: Android 5.1 CPU: Rock chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 2GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K Wi-Fi: 802.11 b/g/n Bluetooth Google Play DLNA Kodi 16.1/ XBMC 3D movie Support ...

Table 9. Refined specification comparison for product ID 20673. Abbreviated for readability.

If we compare line-for-line we can see the product ID 20965 is indeed more similar (almost identical) than that of product ID 21504 when compared to the parent product ID 20673.

Further improvement were attempted by changing the vocabulary to use n amount of words per term instead of each individual word i.e a list of words like [‘os’, ‘version’, ‘android’, ...] becomes [‘os version’, ‘version android’, ...] etc. also ranges from 1 to 3 i.e [‘‘os’, ‘os version’, ‘‘os version android’, …] however this resulted in a significant increase in sparsity by 20%+ as a result the similarity scores dropped too.

Results

As a default for all the models the *analyzer* parameter was set to *“word”* and common english stop words were removed by setting *stop_words* to *“english”*.

To further refine the solution model we removed terms that did not appear in at least two product refined specifications. To achieve this we set the *min_df* parameter to 2 on *TfidfVectorizer*. The *min_df* parameter is defined as “... ignore terms that have a document frequency strictly lower than the given threshold” in the documentation[6]. As a result of this sparsity was reduced making it more easier for our model to determine similarities between products.

119 terms removed after changing the *min_df* parameter:

```
{'1608mhz', 'uhd', 'vcd', 'ddr3', 'hexa', 'x5', 'power', 'trail', '44ghz',
'rk3229', 'a5', 'a7', 'm4a', 'webm', '8g', 'high', 'rk3188', 'rv', 'true',
'v17', 'aarm', 'adapter', '12eu', 'ui', 'specification', 'ra', 'sata3', 'mp1',
't860', 'led', 'air', 'multi', 'v4', 'device', 'performance', 'real', '3368',
's802', 'channels', '128gb', 'rj045', 'source', 'v15', 'peg', 'intel',
'picture', 'xusb', 'option', 'windows', 'isdb', 'rf', 'sata', 'original',
'dtmb', 'cherry', '32', 'powervr', 'gen8lp', 'line', '30fps', 'center', 'atsc',
'format', 'custom', 'keyboard', '30mbps', '2160', 'wireless', 'rtd1295dd', '15',
'oga', 'coax', 'hardware', 'pdif', 'z8300', '60hz', 'ecology', 'hdd', 'mail',
'14', 'flash', '50', 's2', '3840', 'a72', 'adobe', '3218', 'frequency', '30hz',
's805', '905', '45', 'rk3399', 'vmw', 'media', 'mrj45', 'mso9180d1r', '3gpp',
'dvd', '800mhz', '252mhz', 'lch', 'atom', 'mbps', 't2', 'dvb', 'mini', '8tb',
'home', '5mm', '4a', 'ota', 'codec', 'unlicensed', 'file', 'vp6', 'files',
'rch', 'mstar'}
```

[6] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Benchmark vs. Solution Model performance		
	Benchmark Model	**Solution Model**
Text parsed in (seconds):	0.00615	0.27973
Engine trained in (seconds):	0.00234	0.00664
Vocabulary size (word count):	115	240
Sparsity (%):	91.91	62.28

Table 10. Benchmark vs. Solution Model performance.

With regards to computational performance our Solution Model does require more time largely due to the fact as a lot more parsing to the text has to be carried out to extract the required text. This is however something that can be mitigated by pre-parsing the text before hand.

The Benchmark Model has a sparsity of 91.91% for a vocabulary of only 115 terms this means on a product-to-product basis each term holds very little weight which makes it more difficult to calculate similarities between our products. The Solution Model on the other hand has a larger vocabulary of 240 terms but a sparsity of only 62.28% suggesting the bag-of-words produced by our solution model hold more weight from product-to-product.

List of feature words (bag-of-words) used by the Solution Mode:

```
['0ghz', '0v', '10', '1080p', '11', '11a', '11ac', '16', '16gb', '17', '1920x1080',
'1gb', '2160p', '263', '264', '265', '2880', '2880x2160p', '2gb', '2ghz', '2k', '2x',
'2xusb', '3229', '32gb', '3d', '3gb', '3gp', '3x', '3xusb', '400', '400mp', '450',
'450mp', '46ghz', '4g', '4gb', '4ghz', '4k', '4kx2k', '4x', '4xusb', '5g', '5ghz',
'600mhz', '60fps', '64', '64bit', '64gb', '6ghz', '750mhz', '802', '8gb', 'a53', 'a9',
'aac', 'ac', 'ac3', 'acc', 'adpcm', 'airplay', 'amlogic', 'amr', 'android', 'antenna',
'ap', 'ape', 'arm', 'asf', 'audio', 'av', 'avc', 'avi', 'avs', 'band', 'based', 'bit',
'bluetooth', 'bmp', 'book', 'built', 'card', 'channel', 'chip', 'control', 'core',
'cortex', 'cpu', 'cvbs', 'dat', 'dc', 'ddp', 'ddr4', 'decoding', 'decording', 'display',
'divx', 'divx3', 'dlna', 'dts', 'dual', 'dvfs', 'ebook', 'embedded', 'engine',
'external', 'fi', 'fla', 'flac', 'floating', 'flv', 'gb', 'ghz', 'gif', 'google', 'gpu',
'graphic', 'graphics', 'hd', 'hdmi', 'hdmov', 'heaac', 'headphone', 'hevc', 'ieee',
'infra', 'internal', 'ir', 'ism', 'iso', 'jack', 'jpeg', 'jpg', 'kodi', 'lan', 'lc',
'lpcm', 'm4v', 'mali', 'max', 'memory', 'micro', 'microphone', 'mimo', 'miracast',
'mjpeg', 'mkv', 'mouse', 'mov', 'movie', 'mp', 'mp2', 'mp3', 'mp4', 'mpe', 'mpeg',
'mpeg1', 'mpeg2', 'mpeg4', 'mpg', 'nb', 'neon', 'octa', 'ogg', 'optical', 'os', 'otg',
'output', 'pcm', 'pdf', 'penta', 'play', 'playback', 'pmp', 'png', 'point', 'port',
'preinstalled', 'processor', 'prosessor', 'pvr', 'quad', 'ra_cook', 'ram', 'realtek',
'realvideo', 'realvideo8', 'receiver', 'record', 'remote', 'resolution', 'rj45',
'rk3368', 'rm', 'rmvb', 'rock', 'rockchip', 'rtd1295', 'rtk8821', 's905', 's905x',
's912', 'sd', 'shift', 'simd', 'slot', 'slots', 'sp', 'spdif', 'speed', 'support',
'supported', 't820', 't820mp3', 'tf', 'tiff', 'tifff', 'time', 'tp', 'truehd', 'ts',
'txt', 'type', 'unit', 'usb', 'v16', 'v2', 'vc', 'version', 'video', 'vob', 'vp8',
'wav', 'wi', 'wifi', 'wma', 'wmv', 'xbmc', 'xvid', 'zdmc']
```

Benchmark vs. Solution Model Top 10 Terms			
Benchmark Model		**Solution Model**	
Term	**Weight**	**Term**	**Weight**
tv	30.182777	video	8.013830
box	27.390094	4k	7.995391
android	17.244235	memory	7.862221
4k	6.887078	resolution	7.660794
media	5.111881	2k	7.130491
player	4.906306	hd	6.925017
smart	4.584562	32gb	6.455461
best	4.272599	support	5.642847
kodi	3.911404	rm	5.610182
cheap	3.507161	dts	5.324897

Table 11. Benchmark vs. Solution Model Top 10 Terms.

If we look at the top 10 terms selected by the Benchmark Model in **Table 11** we can see none of them can really be used uniquely identify a product apart from the term “4k” which identifies the resolution at which the Tv Box can display video, as not all of them support 4k. Terms like “tv”, “box” and “android” resonate with all the products as this simply just describes what the product is, as a result the Benchmark Model does not return items that share similar specification. We confirm this in **Table 12** by doing the same sanity check using product ID 20673 as we did when defining our Solution Model.

Benchmark vs. Solution model Top selected related item for product ID 20673.			
	Parent product	**Top related item selected by Benchmark Model**	**Top related item selected by Solution Model**
Product ID	20673	20729	20965
Product name	V88 Android TV Box	T95X TV Box	SCISHION V88 Plus Android TV Box
	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play ...	OS Version: Android 6.0 CPU: S905X quad-core cortex-A53 GPU: Mali-450MP Processor Speed (max): 2.0GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 32GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K Wi-Fi: 802.11 b/g/n Google Play Kodi 16.0 / XBMC ….	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 2GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play ...

Table 12. Benchmark vs. Solution model Top selected related item for product ID 20673. Abbreviated for readability.

Conclusion

Free-Form Visualization

For the purpose to illustrate that our Solution Model does produce the expectes results, that is to selected related items that are more so similar than that, that have been manually selected. Two products have been randomly selected, it is then shown that our Solution Model does indeed selected more relevant items and this is further confirmed by running a sanity check on each.

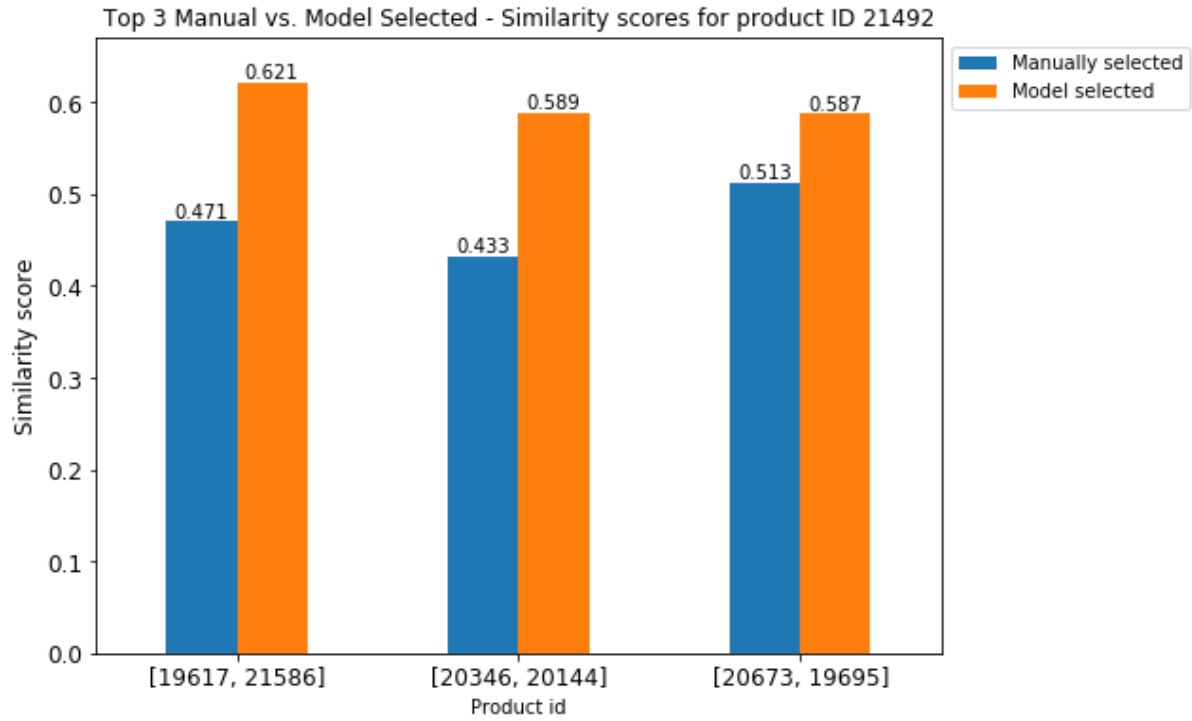


Figure 9. Manual vs. Solution Model selection of related items

Solution model vs Manually top selected related item for product ID 21492.			
	Parent product	**Selected by Solution Model**	**Manually selected**
Product ID	21492	21586	19617
Product name	SCISHION V99 Star TV Box	4K Android TV Box COOWELL V11	MX3 4K Android TV Box
Spec.	OS Version: Android 5.1 CPU: Rockchip 3368 Octa-Core GPU: PowerVR Processor Speed (max): 1.5GHz RAM: 2GB Internal Memory: 16GB External Memory: Support up to 32GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WIFI: 802.11 b/g/n/ac, Dual Band (2.4GHz + 5GHz) Bluetooth Google Play DLNA Kodi / XBMC ...	OS Version: Android 5.1 CPU: Rockchip 3229 Quad-Core GPU: Mali-400 Processor Speed (max): 1.5GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 32GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K Wi-Fi: 802.11 a/b/g/n Google Play Miracast Kodi 16.0v/ XBMC ...	OS Version: Android 4.4 CPU: Amlogic S802 Quad Core GPU: Octa Core ARM Mali-450 up to 600MHz Processor Speed (max): 2.0GHz RAM: 2GB Internal Memory: 8GB External Memory: Support up to 32GB Wi-Fi: 802.11 b/g/n (2.4G/ 5G) Display Resolution(max): 4K x 2K/30fps, 1080P/60fps Support Video Resolution(max): 4K x 2K Bluetooth DLNA, Miracast Google Play XBMC Original ecology and custom UI specification ...

Table 13. Solution model vs Manually top selected related item for product ID 21492

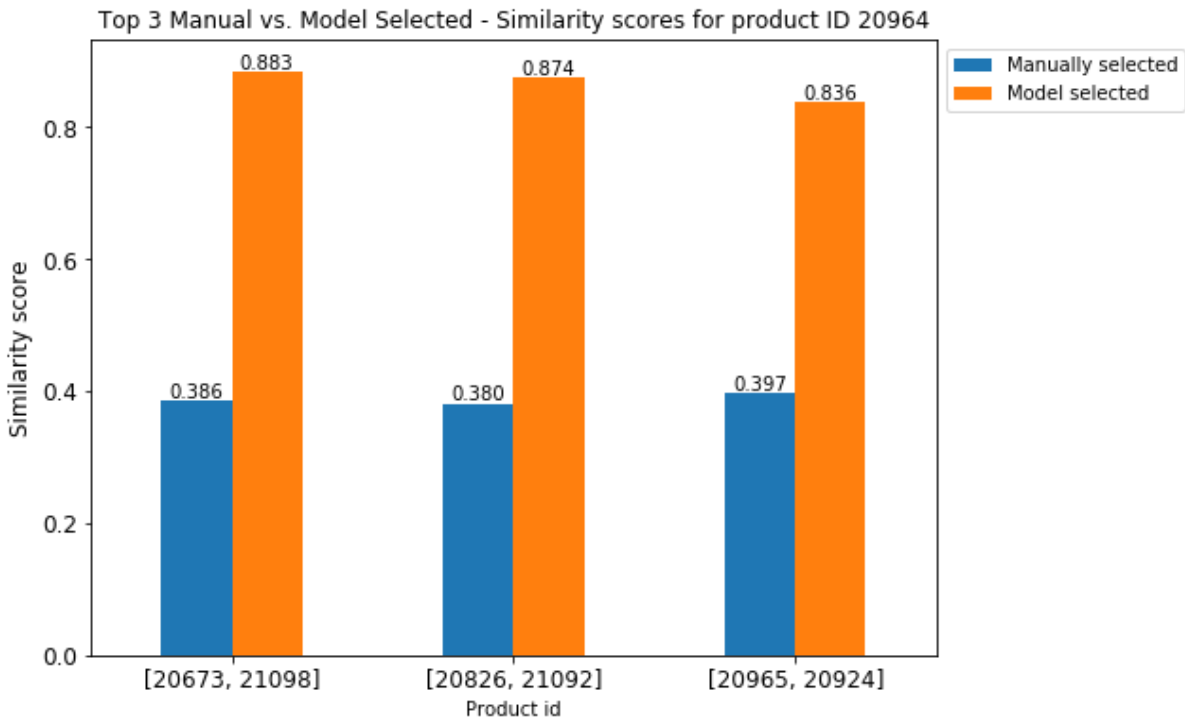


Figure 10. Manual vs. Solution Model selection of related items

Solution model vs Manually top selected related item for product ID 20964.			
	Parent product	**Selected by Solution Model**	**Manually selected**
Product ID	20964	21098	20673
Product name	SCISHION V99 TV Box	4K TV Box	V88 Android TV Box
Spec.	OS Version: Android 6.0 CPU: Amlogic S912 GPU: ARM Mali-T820MP3 Processor Speed (max): 2.0GHz RAM: 2GB Internal Memory: 16GB External Memory: Support up to 32GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Bluetooth 4.0 Google Play XBMC /Kodi: V16.1 DLNA Miracast Airplay ...	OS Version: Android 6.0 CPU: Amlogic S912 GPU: ARM Mali-T820MP3 Processor Speed (max): 1.5GHz RAM: 2GB Internal Memory: 16GB External Memory: Support up to 32GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K Wi-Fi: 802.11 b/g/n, Dual Band (2.4GHz/5GHz) Bluetooth Google Play Kodi V16.1 / XBMC ...	OS Version: Android 5.1 CPU: Rock Chip 3229 Quad-Core GPU: Mali-400MP Processor Speed (max): 1.46GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 16GB Display Resolution: 4K x 2K Supported Video Resolution: 4K x 2K WiFi: 802.11 b/g/n Google Play Kodi 16.1 3D Movie Support ...

Table 14. Solution model vs Manually top selected related item for product ID 20946

Summary

In conclusion the problem to automate the selection of related items, that is preferably better than the manual approach has been achieved. With regards to performance related to our internet solution, as stands now for this dataset of only 65 products we could successfully implement this solution live as it only takes ~0.007 seconds to train our recommendation engine and return the similarity scores and ~0.28 seconds to extract and parse the text that is required. With that said as it is expected for the dataset to grow over time, the better approach is to carry out these task offline and have the results saved into a Database such a Redis for example where these results can easily be queried on demand.

Although our solution model meets the criterias set out, I believe there may still be room for improvement. One of these methods may be to produce our own vocabulary or bag-of-words, however this will require a significantly more parsing of the text and should more categories be added this process would have to be carried out for each, overall this will further add to the computational overhead but is likely to yield even better results.

The biggest challenge is finding the balance between how precise or how well our model should generalize, so when new categories are introduced it deals with them accordingly.

It is also worth mentioning that this approach can further be built on top of to automate the categorization of products by introducing other algorithms.

References

1. Robin Burke | Hybrid Recommender Systems | http://josquin.cs.depaul.edu/~rburke/pubs/burke-umuai02.pdf
2. Aarshay Jain | June 2016 | Quick Guide to Build a Recommendation Engine in Python | https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python
3. Wikipedia | Jan 2017 | Item-item collaborative filtering | https://en.wikipedia.org/wiki/Item-item_collaborative_filtering#cite_note-1
4. Jeffrey D. Ullman | 2014 | Mining of Massive Datasets | Chapter 9 | http://infolab.stanford.edu/~ullman/mmds/ch9.pdf
5. Saleem Ansari | Nov 2015 | Content-based recommendation | https://www.packtpub.com/books/content/content-based-recommendation
6. Michael J. Pazzani & Daniel Billsus | Content-based Recommendation Systems | Chapter 10 | https://www.fxpal.com/publications/content-based-recommendation-systems.pdf
7. Wikipedia | May 2017 | tf-idf | https://en.wikipedia.org/wiki/Tf%E2%80%93idf
8. http://www.recommenderbook.net/media/Recommender_Systems_An_Introduction_Chapter03_Content-based_recommendation.pdf