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 in a single category, based on the content of that product."

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.

Metric

Initially as a simple method before having our final solution model, is to compare the refined specification of a product to its related items and score them according to matching words with the refined specification.

```
# List of words in a product’s refined spec.
['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', 'Wi\xadFi:', '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']

# List of words in a related item’s refined spec.
['OS', 'Version:', 'Android', '4.4', 'CPU:', 'Amlogic', 'S805', 'Quad-Core',
'Cortex-A5', 'GPU:', 'Quad-Core', 'Mali-45', 'Processor', 'Speed', '(max):', '1.6GHz',
'RAM:', '1GB', 'Internal', 'Memory:', '8GB', 'External', 'Memory:', 'Support', 'up',
'to', '32GB', 'Display', 'Resolution(max):1920x1080', 'Support', 'Video',
'Resolution(max):', '4K', 'x', '2K', 'Wi-Fi:', '802.11', 'b/g/n', 'Google', 'Play',
'DLNA', 'Miracast', 'Airplay', 'Kodi', '/', 'XBMC', '16.0', 'HDMI', '4x', 'USB',
'SPDIF', 'SD', 'card', 'slots', 'AV', 'port', 'RJ45', 'port', 'DC', 'IN', 'Video:',
'AVI,', 'RM,', 'RMVB,', 'Ts,', 'VOB,', 'MKV,', 'MOV,', 'ISO,', 'WMV,', 'ASF,', 'FLV,',
'DAT,', 'MPG,', 'MPEG', 'Audio:', 'MP3,', 'WMA,', 'AAC,', 'WAV,', 'OGG,', 'AC3,',
'DDP,', 'TrueHD,', 'DTS,', 'DTS,', 'HD,', 'FLAC,', 'APE', 'Graphic:', 'HD', 'JPEG,',
'BMP,', 'GIF,', 'PNG,', 'TIFF', 'eBook:', 'PDF,', 'TXT']

# Intersection of the two lists, return only words that appear in both lists.
{'SPDIF', '802.11', 'port', 'AV', 'WMA,', 'Graphic:', 'Support', 'x', 'DAT,', 'MPG,',
'FLAC,', '8GB', 'IN', 'CPU:', 'Version:', '(max):', 'ISO,', 'GPU:', 'SD', 'RM,', 'MP3,',
'2K', 'Android', 'Quad-Core', 'to', 'External', 'Speed', 'Memory:', 'AAC,', 'b/g/n',
'MKV,', 'RJ45', 'AVI,', 'DC', 'MOV,', 'RAM:', 'Google', 'Audio:', 'JPEG,', '4K', 'HDMI',
'OS', 'Processor', 'Display', 'Internal', 'Play', 'Video:', 'up', '1GB', 'Kodi',
'Video'}
```

Finally the matching score is then calculated by:
matching score = number of matching words / total number of product spec. words

Thereafter 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]

[2] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

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.

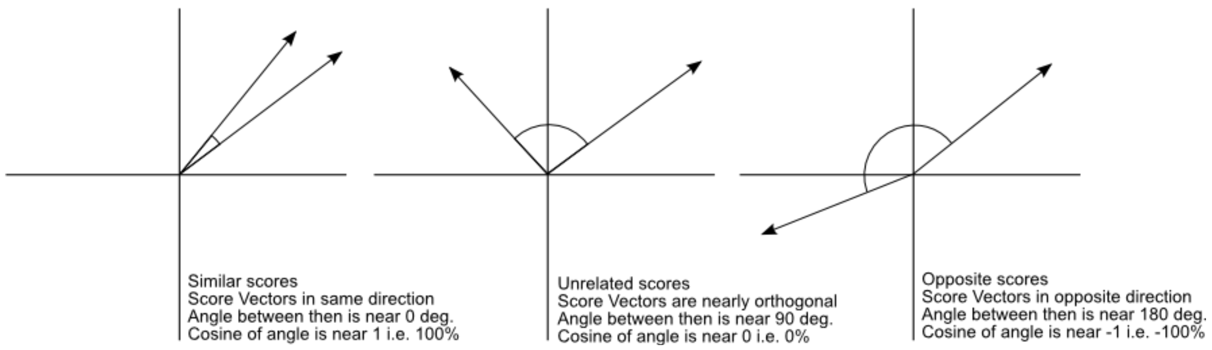

Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

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

Analysis

Data a visual exploration

The dataset we are working with has been provided 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:

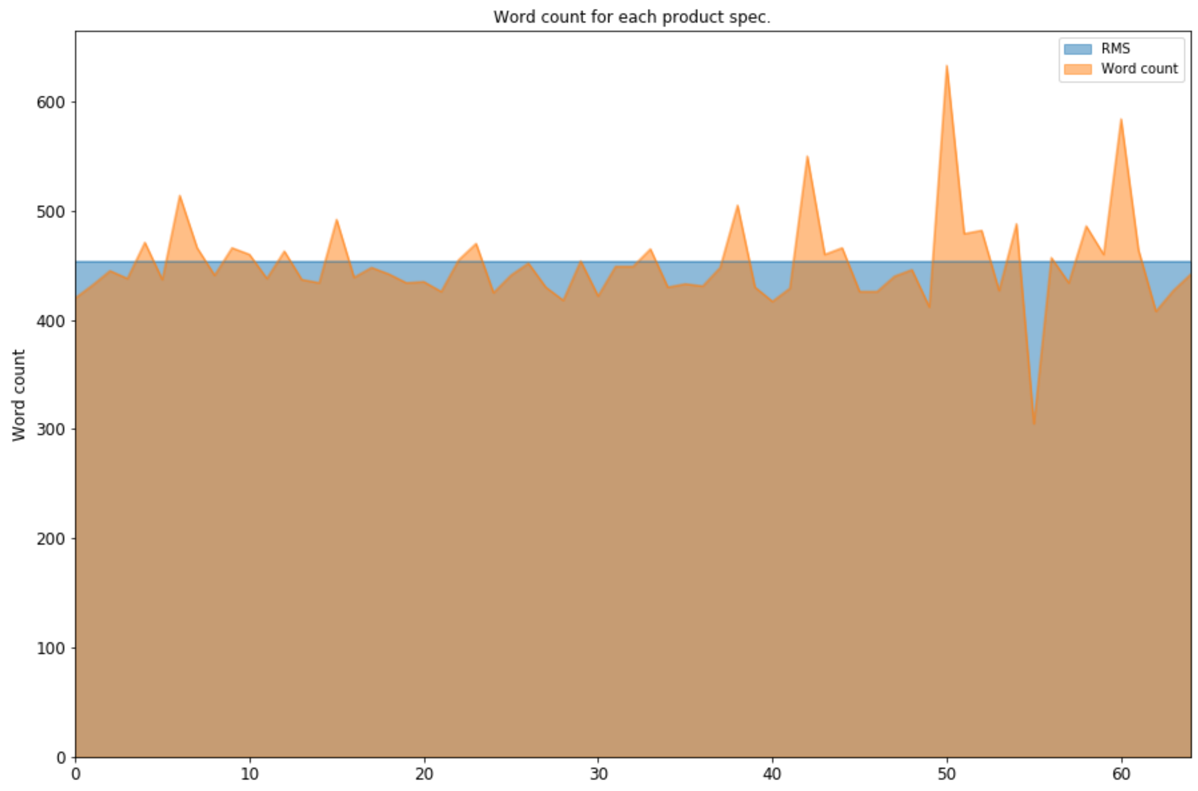
- full and short name
- overview/description
- related products (these have been manually selected)
- specification
- meta keywords and description

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[3] 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

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

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.

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	**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

	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: 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	● 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 ● 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

	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** ● 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	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**

	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: - ליבות 64 ביט 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.

```html
<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**

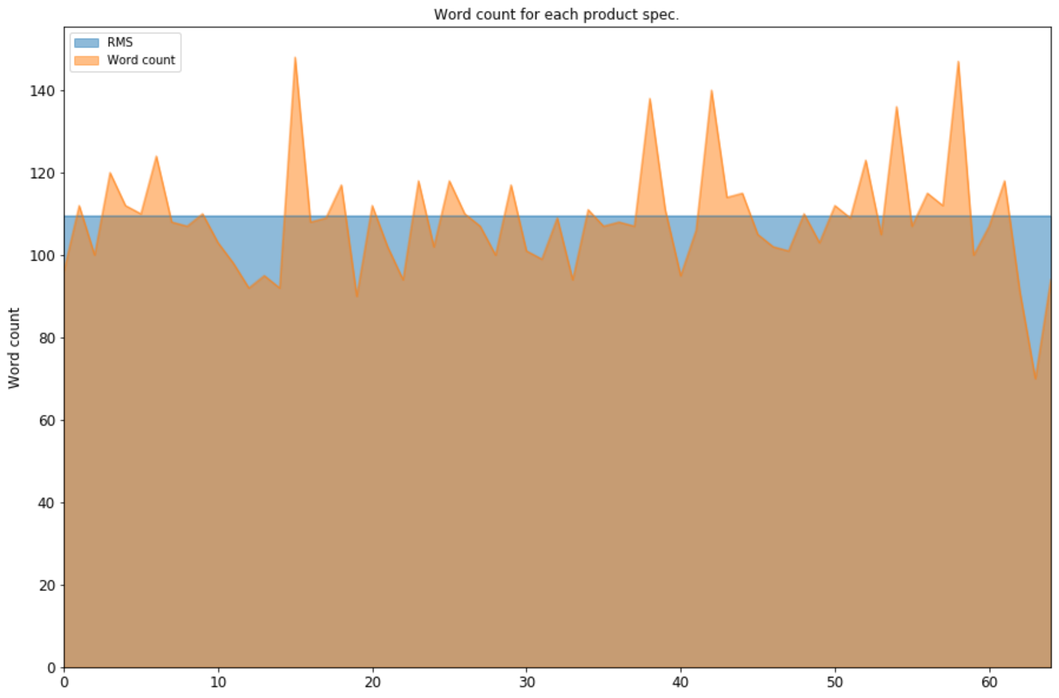
As represented by the pseudo code below we run our loop and confirm that each product does indeed have these headings within their specification.

```
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*[4] which takes in the text data and does both the bag-of-words feature extraction and the *tf–idf* transformation.

Benchmark model

As our benchmark model we selected a product and compared its refined specifications (**General, Ports** and **Media Format** specs. only) to two of its related items specifications that were manually selected then calculated the matching score as illustrated in **Table 2**. below.

Product and related items comparing specifications			
Product ID	20673	19830	20242
Product name	V88 Android TV Box	F7 Android TV Box	MXQ Android TV Box
Refined Spec (text)	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	OS Version: Android 4.4 CPU: Rock Chip 3218 Quad-Core Processor Speed (max): 1.2GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 32GB Display Resolution(max): 1920x1080	OS Version: Android 4.4 CPU: Amlogic S805 Quad-Core Cortex-A5 GPU: Quad-Core Mali-45 Processor Speed (max): 1.6GHz RAM: 1GB Internal Memory: 8GB External Memory: Support up to 32GB Display

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

	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	Wi-Fi: 802.11 b/g/n Bluetooth DLNA Miracast Airplay Google Play Supported XBMC, Kodi AV Out Mini HDMI LAN OTG SPDIF DC IN Micro SD Card Slot 2x USB Video: MP4, 3GP, AVI Audio: MP3, WMA, WAV Graphic: JPG, GIF, PNG, BMP eBook: PDF, TXT	Resolution(max):1920x1080 Support Video Resolution(max): 4K x 2K Wi-Fi: 802.11 b/g/n Google Play DLNA Miracast Airplay Kodi / XBMC 16.0 HDMI 4x USB SPDIF SD card slots AV port RJ45 port DC IN Video: AVI, RM, RMVB, Ts, VOB, MKV, MOV, ISO, WMV, ASF, FLV, DAT, MPG, MPEG Audio: MP3, WMA, AAC, WAV, OGG, AC3, DDP, TrueHD, DTS, DTS, HD, FLAC, APE Graphic: HD JPEG, BMP, GIF, PNG, TIFF eBook: PDF, TXT
	Matching score:	0.438	0.542

Table 2. Comparing parent product to manually selected related items.

We then crosscheck this model with our solution model by checking the matching score as well as similarity score.

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**
<strong> Manufacturer Specifications </strong> <ul id="Manufacturer_Specifications"> </ul>
 <strong> General </strong> <ul id="general"> <li> OS Version: Android 5.1 </li> <li> CPU: Rock Chip 3229 Quad-Core </li> <li> GPU: Mali-400MP	<ul id="general"> <li> OS Version: Android 5.1 </li> <li> CPU: Rock Chip 3229 Quad-Core </li> <li> GPU: Mali-400MP </li> <li> Processor Speed (max): 1.46GHz </li> <li> RAM: 1GB </li> <li> Internal Memory: 8GB	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

</li> <li> Processor Speed (max): 1.46GHz </li> <li> RAM: 1GB </li> <li> Internal Memory: 8GB </li> <li> External Memory: Support up to 16GB </li> <li> Display Resolution: 4K x 2K </li> <li> ...	</li> <li> External Memory: Support up to 16GB </li> <li> Display Resolution: 4K x 2K </li> <li> Supported Video Resolution: 4K x 2K </li> <li> WiFi: 802.11 b/g/n </li> <li> Google Play </li> <li> ...	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 3. The process of parsing the HTML text for a product's specification. HTML has been abbreviated for readability.

Implementation

Process

As illustrated in **Figure 4.** the steps taken to determine the best base 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 area were the product's overview, meta keywords specification and finally the refined specification.

Step 2) As explained in the section **Metrics** we calculate the matching score and similarity score for each product.

Step 3) We then calculate the mean difference between the two sets of scores, this is done to determine if the similarity score being returned form our model is actually consistent with the product's refined specification. We will cover the data in further detail below but as an example should a product have a related item with a matching score of let's say 0.45 and a similarity score of 0.90 this is an indication of inconsistency whereas our model believes that a related item is 90% similar to our parent product where on a spec-to-spec word-for-word comparison the specifications only match for 45% of the content. Ideally what we would like to see is a high similarity score with a high matching score with as small difference as possible.

Step 4) Select the base model that produce the least mean difference.

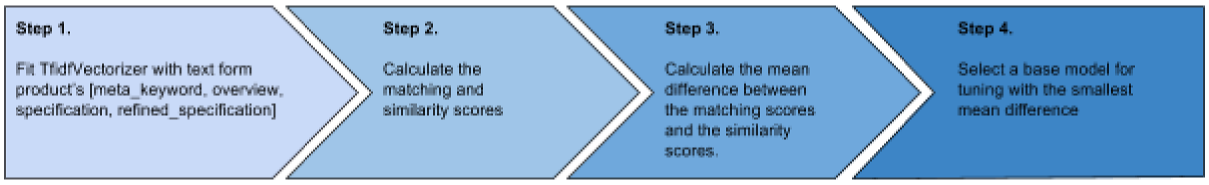

Step 1.
Fit TfidfVectorizer with text form product's [meta_keyword, overview, specification, refined_specification]
Step 2.
Calculate the matching and similarity scores
Step 3.
Calculate the mean difference between the matching scores and the similarity scores.
Step 4.
Select a base model for tuning with the smallest mean difference

Figure 4. Process of finding the base model

Selecting the base model

Continuing on we will explore the data that was used to determine the best base model and how it compares to our benchmark model using our parent product with ID 20673.

Model 1. Using product's meta keywords:

Related items scores based on meta keywords. (Parent product: 20673)							
	Top 5 products selected by model					Benchmark (Manually selected product)	
Product ID	20729	20781	20924	20656	20578	**19830**	**20242**
Similarity score	0.952	0.913	0.895	0.869	0.853	**0.672**	**0.789**
Matching score	0.552	0.563	0.594	0.552	0.583	**0.438**	**0.542**
Mean difference	**0.328**					**N/A**	

Table 4. Related items scores based on meta keywords.

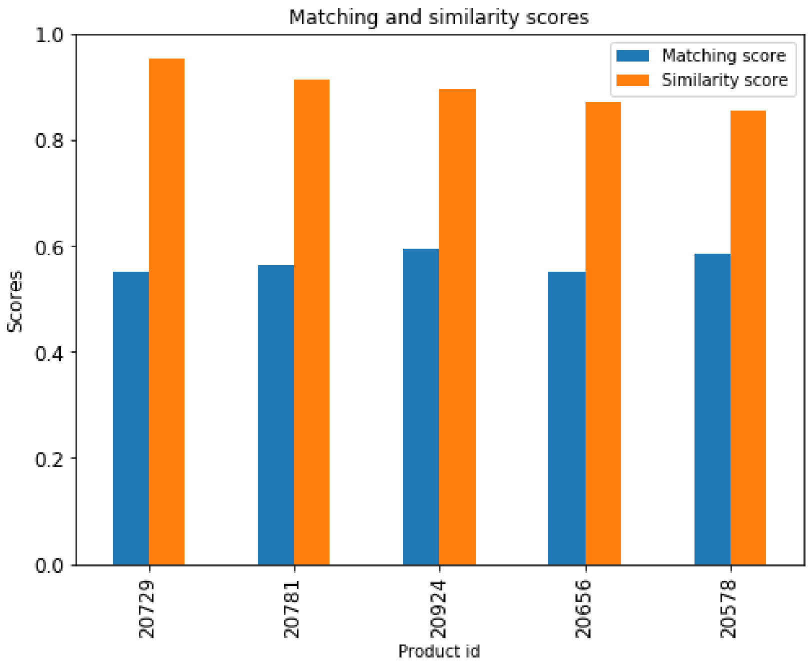


Figure 5. Matching and similarity scores based on meta keywords.

Key points:

- Model similarity and matching score are inconsistent. The product with the highest matching score (20924) of 59% only received a similarity score of 90%. However the product with the highest similarity score (20729) of 95% only received a matching score of 55%.
- On average the difference between the two sets of scores is 33%
- Model still recommends more similar products when compared to the benchmark model.

Model 2. Using product's overview

Related items scores based on product overview. (Parent product: 20673)							
	Top 5 products selected by model					Benchmark (Manually selected product)	
Product ID	20965	20346	20826	20144	20578	**19830**	**20242**
Similarity score	0.424	0.395	0.395	0.347	0.334	**0.240**	**0.323**
Matching score	0.854	0.552	0.781	0.635	0.583	**0.438**	**0.542**
Mean difference	**0.302**					**N/A**	

Table 5. Related items scores based on products overview.

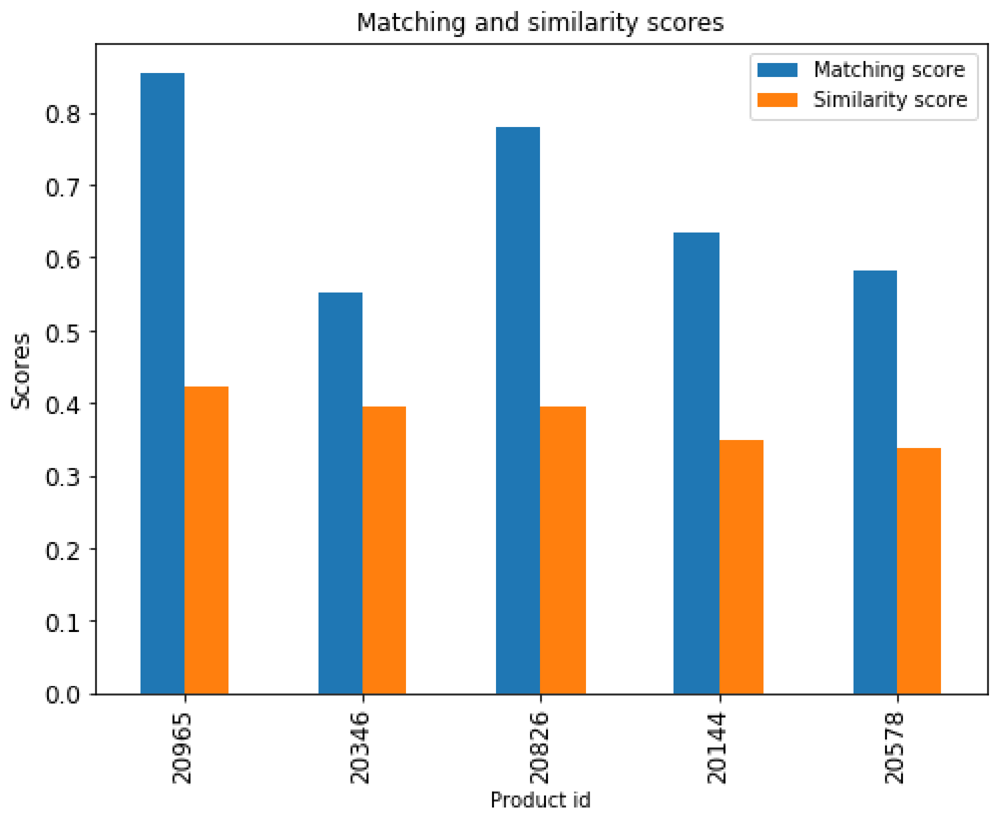


Figure 6. Matching and similarity scores based on product overview.

Key points:

- Inconsistency between scores with slight improvements over **Model 1**.
- Similarity scores a relatively low which is expected as we would prefer each products **overview to be unique** in comparison to others. As a result even though it improves slightly from **Model 1**, it doesn't provide a sound model when looking for similar items and we could rule this approach out even though it beats our Benchmark model.

Model 3. Using product's specification

Related items scores based on product specification. (Parent product: 20673)							
	Top 5 products selected by model					**Benchmark (Manually selected product)**	
Product ID	21503	21504	20846	20965	21295	**19830**	**20242**
Similarity score	0.889	0.879	0.836	0.825	0.795	**0.653**	**0.645**
Matching score	0.771	0.781	0.656	0.854	0.552	**0.438**	**0.542**
Mean difference	**0.122**					**N/A**	

Table 6. Related items scores based on products specification.

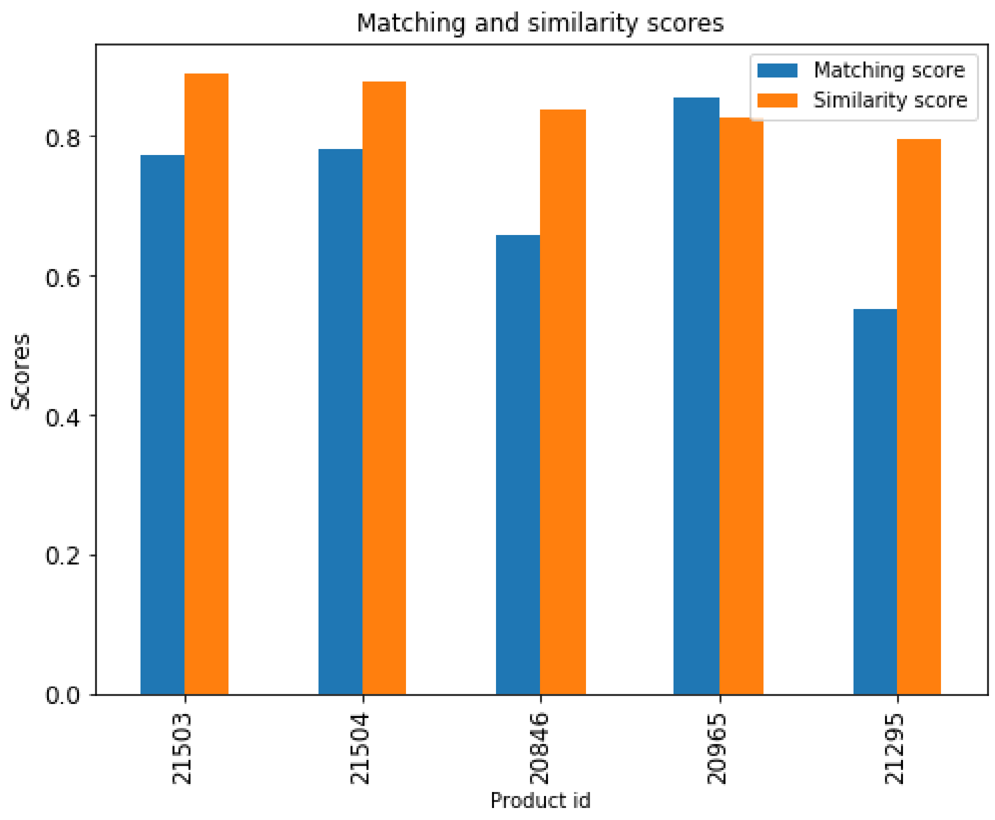


Figure 7. Matching and similarity scores based on product specification.

Key point:

- We still see inconsistency in the scores where ideally we would like to see product 20965 receiving the highest of both sets of scores.
- The difference between the two sets of score average only 12%.
- Using product specifications provides us with more consistent text data where the structure and attributes are almost the same from product to product, forming a solid platform when looking for similar items.

Model 4. Using product's refined specification

Related items scores based on product's refined specification. (Parent product: 20673)							
	Top 5 products selected by model					**Benchmark (Manually selected product)**	
Product ID	21504	21503	20965	20826	21587	**19830**	**20242**
Similarity score	0.928	0.919	0.895	0.776	0.694	**0.371**	**0.354**
Matching score	0.781	0.771	0.854	0.781	0.667	**0.438**	**0.542**
Mean difference	**0.072**					**N/A**	

Table 7. Related items scores based on product's refined specification.

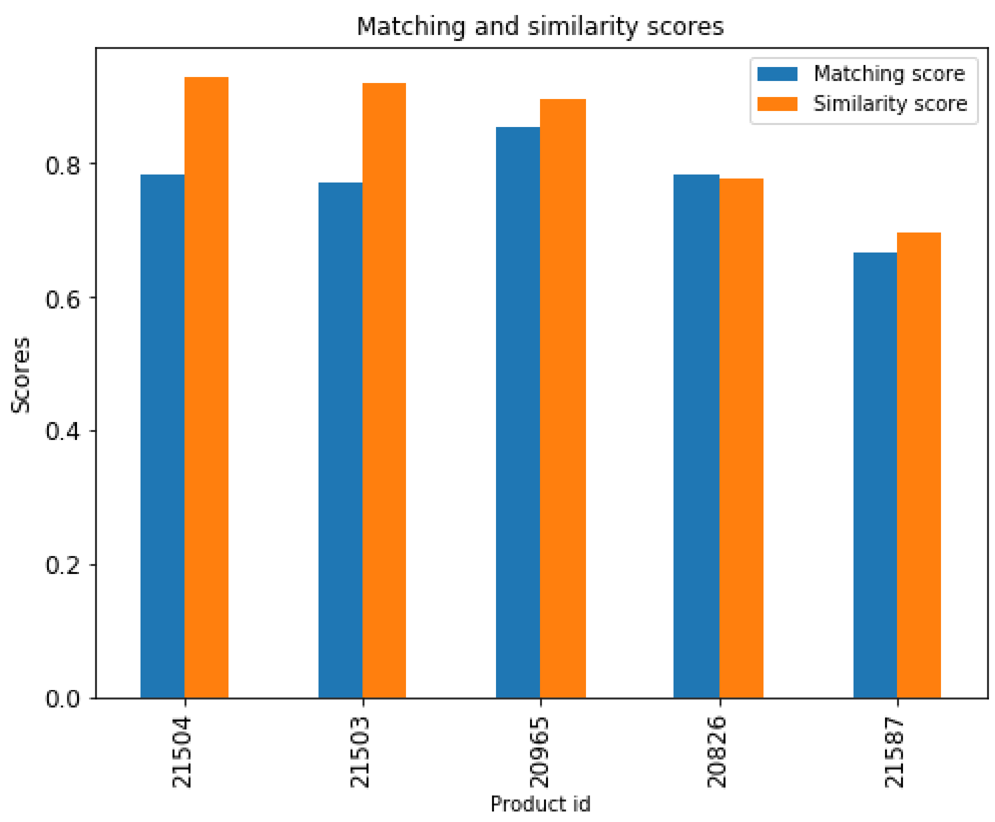


Figure 8. Matching and similarity scores based on product's refined specification.

Key point:

- Again there is still inconsistency in the scores.
- The difference between the two sets of score average only 7%.
- By using the refined specification we only concern ourselves with the product attributes that are most important making this Model a good Base Model.

In conclusion all our Models were able to recommend items that were more similar than that of our benchmark model where the related products were manually selected. The issue with inconsistency in the scores whereas the product with the highest matching score isn't necessary the one with the highest similarity score can be improved on by further tweaking our model which is discussed in the next section.

Refinement

Although our base model outperforms the benchmark model and is selecting items more relevant it would be ideal if we could tune our model to at least select the top 3 related items most of the time.

As an initial step to refine our 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[5] from 359 words to 240.

Similarity scores after removing words. (Parent product: 20673)					
	Top 5 products selected by model				
Product ID	20965	21504	21503	20826	21587
Similarity score	0.986	0.9281	0.919	0.776	0.694
Matching score	0.854	0.781	0.771	0.781	0.667
Performance	0.00765 seconds to train				
Vocabulary	240 words				

Table 8. Similarity scores after removing words.

Looking at **Table 8.** we can see that the top 3 products are being recommended and the scoring sets are somewhat more consistent. To see how well our model can generalize let's try using another two different products as the parent.

Similarity scores after removing words. (Parent product: 21492)

[5] Vocabulary is simply a list of words selected by the tf-idf algorithm.

	Top 5 products selected by model				
Product ID	21586	20144	19695	20892	20891
Similarity score	0.621	0.589	0.587	0.571	0.562
Matching score	0.564	0.574	0.564	0.594	0.574
Performance	0.00765 seconds to train				
Vocabulary	240 words				

Table 9. Similarity scores after removing words.

Similarity scores after removing words. (Parent product: 20964)					
	Top 5 products selected by model				
Product ID	21098	21092	20924	21099	21297
Similarity score	0.883	0.874	0.836	0.831	0.814
Matching score	0.813	0.813	0.794	0.721	0.757
Performance	0.00765 seconds to train				
Vocabulary	240 words				

Table 10. Similarity scores after removing words.

Given that each specifications attribute usually takes the form of <attribute> : <value> further tweaking was made to the model by changing the vocabulary to use paired words instead of each individual word i.e a list of words like [‘os’, ‘version’, ‘android’, ...] becomes [‘os version’, ‘version android’, ...] etc. Let us revisit the results after these changes.

Similarity scores after removing words. (Parent product: 20673)					
	Top 5 products selected by model				
Product ID	20965	21504	21503	20826	21587
Similarity score	0.975	0.841	0.829	0.627	0.534
Matching score	0.854	0.781	0.770	0.781	0.667
Performance	0.01069 seconds to train				
Vocabulary	529 words				

Table 11. Similarity scores after removing words.

Similarity scores after removing words. (Parent product: 21492)

	Top 5 products selected by model				
Product ID	21503	21504	20846	21586	21587
Similarity score	0.503	0.501	0.470	0.468	0.446
Matching score	0.594	0.603	0.594	0.564	0.574
Performance	0.01069 seconds to train				
Vocabulary	529 words				

Table 12. Similarity scores after removing words.

Similarity scores after removing words. (Parent product: 20964)					
	Top 5 products selected by model				
Product ID	21098	21092	20924	21099	21297
Similarity score	0.748	0.680	0.649	0.626	0.620
Matching score	0.813	0.813	0.794	0.721	0.757
Performance	0.01069 seconds to train				
Vocabulary	529 words				

Table 13. Similarity scores after removing words.

Although the similarity scores have dropped slightly we now see that in all the above cases the top 3 products are mostly selected. Comparing **Table 12** and **Table 9** where the parent product is **21492** we can see a different set of products have been ranked where the top 3 have higher matching scores than that chosen by our previous model. Again sticking with product **21492** as the parent product we can confirm our model is indeed selecting more relevant products than the manual approach as illustrated in the **Table 14** below.

Manual vs. model selection of related items. (Parent product: 20964)						
	Related items selected by solution model.			**Related items manually selected.**		
Product ID	21503	21504	20846	**19617**	**20346**	**20673**
Similarity score	0.503	0.501	0.470	**0.234**	**0.152**	**0.425**
Matching score	0.594	0.603	0.594	**0.475**	**0.554**	**0.594**

Table 14. Manual vs.solution model selection of related items.

Conclusion

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 ~0.011 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, and 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 off 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