Can Machines Learn to Map Creative Videos to Marketing Campaigns?

Jarod Wang
Amazon
donghaw@amazon.com

Chirag Mandaviya
Amazon
mandac@amazon.com

Abstract

The demand for accurate estimation of marketing's incremental effect is rapidly increasing to enable marketers make informed decisions on their ad investment. The process of ad-mapping links an ad shown to consumers on the fixed marketing channels (Linear TV, Digital, Social) to a marketing creative video. Thus, an accurate ad-mapping, which is a special case of video copy detection, is a cornerstone of ensuring exposure of ad is linked to the correct creative and marketing campaign and hence precise marketing effect measurement. With each campaign having tens of creatives and each country (marketplace) having tens of marketing campaigns each week, the current process of human annotation of hundreds of creatives requires over 800+ team's hours annually. Moreover, this manual process causes significant challenges in onboarding new businesses and countries to measurement due to the absence of intelligent model based ad-mapping solution. To solve this problem, we built a machine learning (ML) model that leverages fingerprinting methodology and automatic language identification technology to match each creative to the marketing campaign. In the paper, we present the computing algorithm and implementation details with results from actual campaign dataset. Extensive validation and comparison studies conducted demonstrates improved mapping results with the new proposed method, achieving 87% F1 score and 82% accuracy. To our best knowledge, this is the first model that uses a fusion of visual, audio, language and metadata features for such ML based content mapping solution. The proposed method leads to 90% reduction on the time spent on ad-mapping compared to manual solutions.

1. Introduction

Marketing measurement provides insights to marketers to optimize their campaign budget and portfolio planning. To measure the campaign effectiveness and efficiency, we need to understand the exposure of the campaign ad to estimate its incremental effect. There are two independent entities involved. The marketing teams create the ads, assign metadata such as brand, product and campaign, and distribute them through media buying agencies for execution. Ad tracking system tracks the viewership or exposure of the ads on marketing channels. If the content created and the one tracked were identical, mapping would be easy. In the real world, the content takes different forms based on the marketing channel, device type, publishers, data collection process, etc. This results in multiple changes in the original creative as shown in Figure 2, yielding two problems: 1) the same creative shows up with multiple names and 2) creatives are incorrectly mapped to products and campaign names thus leading to incorrect attribution.

The challenge is to find the most similar video in the reference videos library for a query video. Without an automated workflow, reference videos need to be mapped on a regular (typically weekly) basis. In a simplified workflow (Figure 1), post receiving a list of ads in the viewership data, the product and campaign information is assigned either using historical mapping for ads seen before, or watching videos of new ads on both sides. It can take several hours every week to process 100+ ads in each country for one marketing channel (e.g., Linear TV). This manual process poses severe challenges in scaling the measurement solution to different countries due to the combination of multiple ads, campaigns and increased time for each mapping.

Our goal is thus to create a fully automated solution that helps to map creative videos to a given campaign. In this paper, we propose a machine learning (ML) based approach leveraging compute vision algorithms, audio fingerprint algorithms, speech and language detection to automate this challenging ad-mapping problem. The main contribution of the paper is two-fold: We show that the ML based approach is effective in ad-mapping automation, achieving 78% F1 score and 71% accuracy using image feature alone. Combination of image, audio, video and language features could be leveraged to further augment the baseline model (based on only image features) increasing F1 score to 87% and accuracy to 82%.

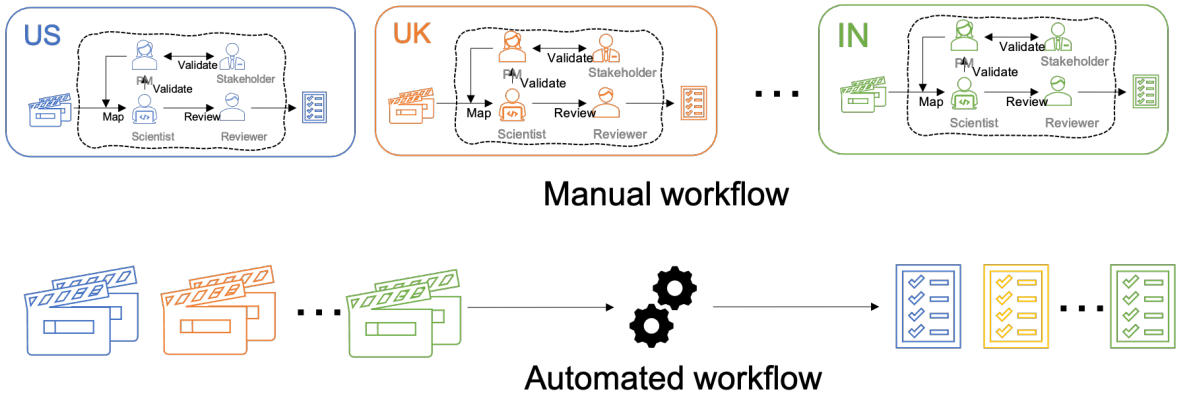
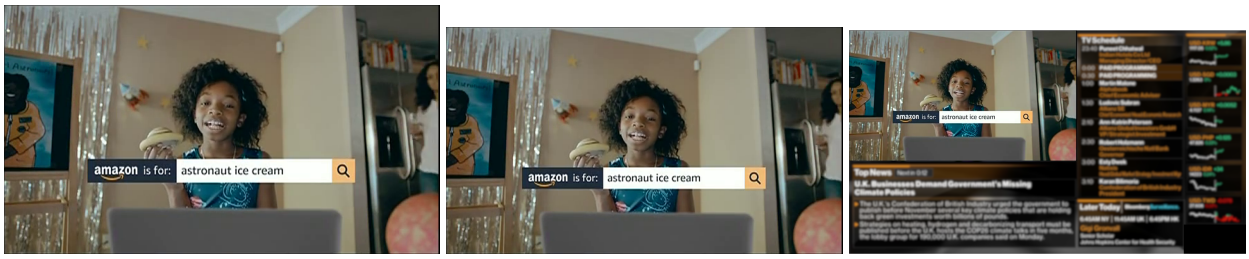
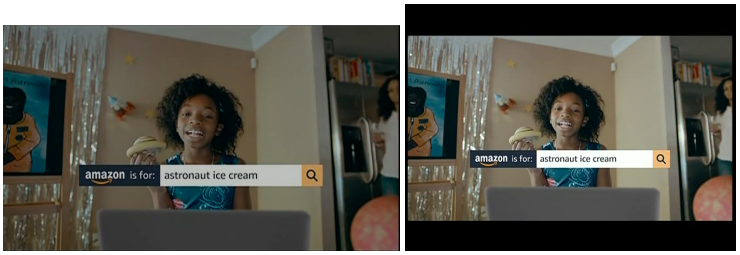


Figure 1: The manual workflow shows that each country requires additional resources to map ads, increasing the latency and the operations cost. The proposed automation solution simplifies the ad-mapping workflow, making it scalable to multiple marketing channels and countries.



(a) Original, 1920 x 1080 pixels (b) 960 x 540 pixels (c) Embedded



(d) 75% brightness (e) Letter box

Figure 2: Illustration of potential changes to the original video.

2. Related work

Ad-mapping is a special case of video copy detection. A video copy is a transformation of a video by means of inserting text, changing resolution, cropping, letter-box, change of gamma, etc. Perceptual hashing or content-based identification is one of the most popular solutions [7]. The created hash code are fingerprints, which are a vector of numbers typically binaries. The goal of the video fingerprinting is to identify a given query video in a reference video database. This field attracts a lot of attention. For example, [9] extracts video fingerprints based on the centroid of gradient orientations. [10] proposes a randomly projected binary features for fast similarity computation. Both methods use visual features. In this paper, we build on top of the previous work [6]. We present novel framework of adding video duration, audio and language features, training 8 ML models, testing on the latest data, and integrating the solution with the data pipeline in production.

3. Dataset

We use a real world dataset of creative videos. The sampled dataset consists of a reference set of 34 videos received from creative teams. And a query set of 61 videos received from the ad tracking system. Table 1 describes the basic statistics of the dataset. The objective is to map the 61 query videos to the reference set from the creative team.

In other words, we need the N-to-1 mapping from the query set to the reference set. We manually label the data to obtain the mapping. The videos are in English or Spanish. There are videos with the same visual content but in different languages. The video duration falls into three groups: 15, 30, and 60 seconds campaigns.

4. Methods

The proposed solution builds on top of various compute vision [5], audio fingerprint [8], and speech recognition [3] algorithms. The goal is to learn a function that measures similarity of two input videos and thus determine the mapping. The features include image fingerprint, video duration, audio and language of the content (videos).

Formally, we want to find the most similar video in reference videos for a query video. The reference videos are creatives produced by marketing teams. Denote the set as V. We represent a video in the set $v \in V$ using features such as video duration, image fingerprint, language, and audio. Denote $v = (f^V(v), f^I(v), f^L(v), f^A(v))$ respectively. The query video $q = (f^V(q), f^I(q), f^L(q), f^A(q))$ is from ad tracking system.

We formulate the problem as a binary classification problem. For each pair of a query video and reference video, we have features, i.e $\{x_{i,v}, x_{i,q}\}$. The dependent variable y_i is 1 if the query video matches the reference video, 0 otherwise. Denote $y_i \in \{0, 1\}$. The objective is to learn a similarity function $s(x_{i,v}, x_{i,q})$.

$$y_i = s(x_{i,v}, x_{i,q}) \tag{1}$$

5. Features

We extract image fingerprint, duration, language, and audio from each video. Since we are interested in finding the most similar video, we create features in pairs: one is a query video and the other is a reference video. Table 2 shows an example of raw features. Below describes how to extract the features and derives additional information.

5.1. Video duration

Video duration can be directly computed from the video, i.e. fps $\times$ number of frames. We create a feature that represents the difference of duration between two videos by taking the difference.

5.2. Image feature

We sample frames from a video controlled by the frame-per-second (fps) parameter. Each frame is represented by a vector of 64-bit hash code. E.g. $f^I(v_i) = (0011\cdots, 0010\cdots, \cdots)$ using four image fingerprint algorithms. We then compare all frames from video A to all frames from video B, find the most similar pairs with respect to Hamming distance, and count the number of "best matching" frames. Best matching means two frames have Hamming distance less than 5.

The four image fingerprint algorithms consist of three steps to compute a 64-bit hash for an image: grayscale, scaling down, transformation. Table 3 summarizes each algorithm. [5] provides detailed discussion on the hash functions.

How to calculate the %matched? Consider two videos A and B in Table 4. Video A consists of 3 frames and B consists of 4. The values in the table represents the Hamming distance. The last column shows the minimum distance between A and B. In the example, we see Frame 2 of Video A is similar with Frame 3 of Video B with 1 bit difference. The Hamming distance is less than 5 so it is a match. Similarly, Frame 3 of Video A matches Frame 4 of Video B. Overall, Video A has 2 matching frames. The % matched is 66.67% (Video A as the denominator).

5.3. Audio feature

As we represent each video frame with a 64-bit fingerprint, we lose information. It makes it hard to distinguish two distinct videos with similar background using image fingerprint alone. To solve this problem, we augment the model with the audio features. The idea is similar to image fingerprint. We generate audio fingerprints for each video and compare the distance of two fingerprints to determine the similarity.

We use Chromaprint [8] to generate the audio fingerprint. Chromaprint represents an audio as a spectrogram, which shows how the intensity of frequencies of the audio changes over time. It then transforms frequencies to "chroma features" using the system developed in [4]. The transformation makes the representation robust to changes caused by codec. To create the fingerprint, it moves a sliding window through the spectorgram, applying 16 filters that capture intensity differences across chroma features and time to form a 32-bit integer.

We follow the method described in [1] to determine similarity. For each audio pair: query and reference, it computes the Hamming distance and normalizes the result. It also handles the offset in audio files, where the query and reference audios are shifted at the start or end. The outcome is a number ranging from 0 (least similar) to 1 (most similar).

5.4. Language feature

We face a challenge that a creative video has multiple language versions. We detect the language using Amazon Transcribe [3]. It outputs 5 detected languages ranked by the confidence score. For example (ja-JP, 0.2461), (en-US, 0.233), (en-GB, 0.2287), (de-DE, 0.1553), (en-AU, 0.1368).

Data source	Video count	Size (MB)				Count by language	
		Total	Mean	Min	Max	English	Spanish
Reference	34	32,103	892	15	10,876	27	7
Query	61	297	5	1	15	51	10

Table 1: Dataset description

Reference	Duration(q)	Duration(r)	%matched	Language(q)	Language(r)	Audio similarity
a1	15	15	0.8	EN	EN	0.7
a2	15	15	0.7	EN	ES	0.5
a3	15	30	0.5	EN	EN	0.6

Table 2: The table illustrates features. We have one query video, t1, and three reference videos: a1, a2 and a3. The letter in the parenthesis represents the query video (q) or the reference video (r). %matched = No. of matching frames / No. of frames in the query video.

We do not distinguish British English vs American English, so we merge the results by the main language. In the example, the merged result is (en, 0.5985), (ja-JP, 0.2461), (de-DE, 0.1553). And the detected language is English. We then create a binary feature to indicate if two videos have the same language in Equation 2. If we are not able to determine the language we set the value to 0.

$$\text{Lan}(x_i^v, x_i^q) = 1 \quad \text{if} \quad f^L(x_i^v) = f^L(x_i^q) \tag{2}$$
$$= 0 \quad \text{o.w.} \tag{3}$$

In our experiment, we use the Amazon Transcribe to automatically detect the language. In practice, the ad tracking system can provide the language for each video leading to higher language accuracy. Here we want to demonstrate when language information is not available we can automatically detect the language to improve the model performance.

We find that Amazon Transcribe is 95% accurate on a dataset of 75 videos. One caveat is that it cannot detect the language if the speech segment of the video is too short.

6. Results

We evaluate the models on a real world dataset described in Section 3. In the first experiment, we want to find the best performing model among 8 popular ML models. Results show that Extra Trees performs the best. In the second experiment, we test various combination of the features using Extra Trees. We use 5 fold cross validation in both experiments. The three metrics are F1-score, accuracy and ROC-AUC.

6.1. Selecting ML model

We compare the performance on 8 ML models. The two parametric models are Logistic regression (logistic_regression) and Gaussian Naive Bayes (gaussian_nb). The six non-parametric models are Multi-Layer Perceptron (MLP), ADA boost (ada_boost), random forest (random_forest), extra trees (extra_trees), k-nearest neighbors (k_neighbors), and XGboost (xgboost). Again we evaluate each model on a 5 fold cross validation setting. We do not tune hyperparameter and instead use the default values with changes in 2 models: in extra trees model, we give the positive twice the weight as the negative one, in k-nearest neighbors, we set k to 1. Figure 3 shows that extra trees model outperforms the others.

6.2. Feature comparison

We use the model trained on image feature only as the baseline model. We then add video duration, language feature, audio feature, and their combinations and repeat the experiment.

Table 5 shows the performance of 6 models with different feature sets. Comparing to the baseline, model M6 improves F1 and accuracy to 87% (up from 78%) and 82% (up from 71%) respectively.

In Figure 4, each box + whisker shows the scores from a 5 fold cross validation run. The box shows the quantiles of the score while the whiskers extend to show the rest of the distribution. Points outside the box are "outliers". We group the boxes based on the finger print algorithm in the X-axis. The first four groups represent the average, perpetual, wavelet, and difference algorithm, respectively. The last column shows the results of combining them.

Within each group we clearly see that adding duration, language, and audio features yields the highest accuracy. Combining all four features constantly yields the best score across the groups.

Table 6 shows the 3 metrics for each algorithm and feature combination averaging over 5-fold cross validation

	Average	Difference	Perpetual	Wavelet
Grayscale	Yes	Yes	Yes	Yes
Scale down (pixels)	8×8	9×8	32×32	8×8
Transform	Compute the average of all gray values of the image. Exam individual pixels.	For each row, the first 8 pixels are compared to their neighbor to the right.	Apply discrete cosine transform (DCT) by row Apply discrete cosine transform (DCT) by row Crop upper left 8×8 pixels Compare to median of gray values of the image	Apply wavelet transform 4 times Compare to the median of gray values of the image

Table 3: Four image fingerprint algorithms

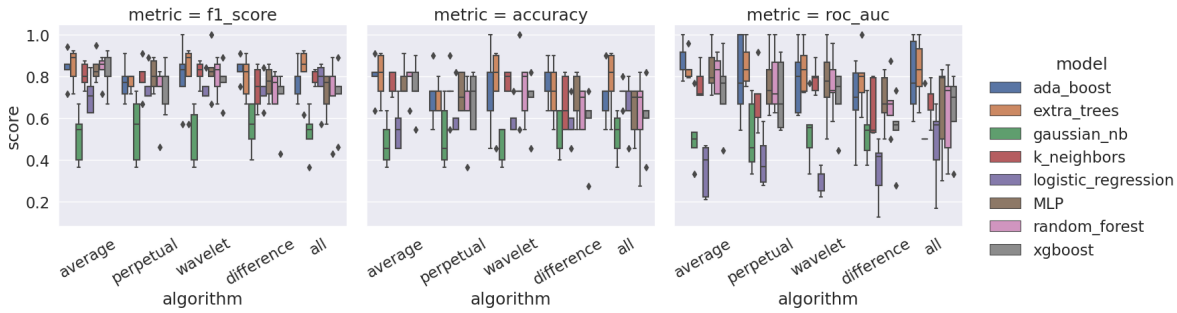


Figure 3: Performance comparison by ML models. Extra Trees model in orange outperforms the other models.

runs.

7. Conclusion and future work

Through the application of the ML model using video duration, image fingerprint, audio fingerprint and language features, we successfully solved the problem of automatically mapping ads to creative videos. The model achieves 82% accuracy on a real world dataset. With the ML model implemented in production in the measurement system, the automated solution would save 90% of time currently spent on the ad-mapping process, enabling the team to scale the measurement solution to more businesses and geographies. This will also greatly facilitate onboarding of new ad solutions.

As for future work, we plan to improve the mapping accuracy by adding text features. First, some of the videos are mainly a few lines of text on a black background. As the image fingerprint algorithms scale down the image, we lose the text information. One way to solve the problem is to detect the text in the video. Amazon AWS's Rekognition [2] which provides text recognition service could be used to further enhance the model with the text identification. We can then convert text to features using methods such as bag of words, TFIDF, word2vec.

Second, the current pair wise comparison has high computation cost on large datasets. Since we compare every frame from a query video to every frame from each and every reference video, the total number of comparison is $900 \times$ No. of query videos $\times$ No. of reference videos (assuming a video has 30 frames at 1 fps sample rate). This is not a key issue for now because in the current sampled dataset the number of videos is small (in hundreds). Once the image fingerprint is computed, the computation finishes in ¡20 minutes. We can reduce the time so the solution can scale to large datasets. The key idea is that many neighboring frames are very similar, so we can keep the key frames and discard the rest.

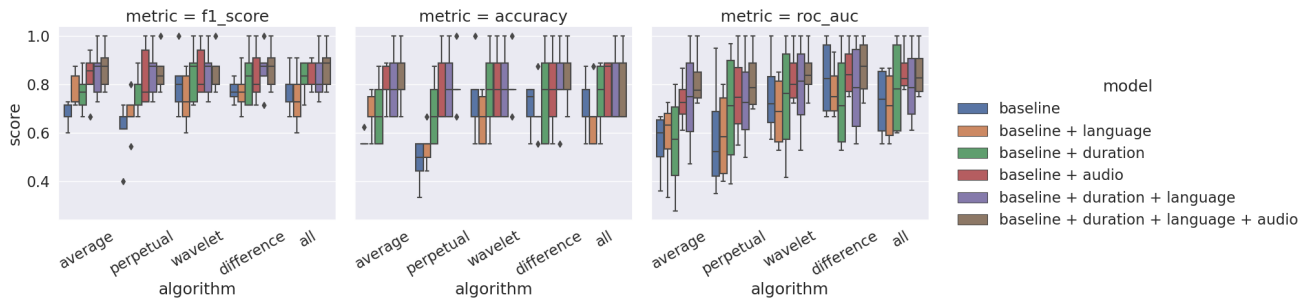


Figure 4: Performance comparison of 6 models with different feature sets.

References

[1] Shivam Aggarwal. Audio signals: Comparison, Mar 2017.

[2] Amazon. Amazon rekognition: Automate your image and video analysis with machine learning., 2021.

[3] Amazon. Amazon transcribe: Automatically convert speech to text, 2021.

[4] M.A. Bartsch and G.H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[5] Content Blockchain. Testing different image hash functions.

[6] Amit Gupta. Machine learning solution for creative mapping.

[7] Ton Kalker, Jaap Haitsma, and Job C. Oostveen. Issues with digital watermarking and perceptual hashing. In Andrew G. Tescher, Bhaskaran Vasudev, and V. Michael Bove Jr., editors, *Multimedia Systems and Applications IV*, volume 4518, pages 189 – 197. International Society for Optics and Photonics, SPIE, 2001.

[8] Lukáš Lalinský. How does chromaprint work?, Jan 2011.

[9] Sunil Lee and Chang D. Yoo. Robust video fingerprinting for content-based video identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(7):983–988, 2008.

[10] Chenxia Wu, Jianke Zhu, and Jiemi Zhang. A content-based video copy detection method with randomly projected binary features. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 21–26, 2012.

	B, Frame 1	B, Frame 2	B, Frame 3	B, Frame 4	Min distance
A, Frame 1	27	6	7	17	6
A, Frame 2	8	10	1	3	1
A, Frame 3	14	19	30	4	4

Table 4: An example of Hamming distance between two videos. Rows represent frames in Video A; columns represent frames of Video B.

Model	Feature	F1		Accuracy		ROC_AUC	
		mean	std	mean	std	mean	std
M1	baseline	0.78	0.09	0.71	0.12	0.72	0.16
M2	baseline + language	0.74	0.12	0.66	0.13	0.71	0.16
M3	baseline + duration	0.83	0.07	0.75	0.14	0.79	0.21
M4	baseline + audio	0.85	0.06	0.8	0.12	**0.86**	0.10
M5	baseline + duration + language	0.84	0.11	0.78	0.16	0.80	0.17
M6	baseline + duration + language + audio	**0.87**	0.09	**0.82**	0.15	0.85	0.11

Table 5: Adding duration, language, and audio feature improve the performance.

		F1		Accuracy		ROC_AUC	
Algorithm	Features	mean	std	mean	std	mean	std
average	baseline	0.68	0.05	0.57	0.03	0.56	0.14
average	baseline + language	0.78	0.07	0.68	0.09	0.58	0.17
average	baseline + duration	0.77	0.08	0.69	0.14	0.56	0.23
average	baseline + audio	0.83	0.11	0.80	0.09	0.73	0.11
average	baseline + duration + language	0.85	0.11	0.80	0.14	0.74	0.23
average	baseline + duration + language + audio	0.87	0.09	0.82	0.13	0.82	0.13
difference	baseline	0.77	0.05	0.71	0.1	0.83	0.17
difference	baseline + language	0.77	0.09	0.69	0.12	0.78	0.12
difference	baseline + duration	0.83	0.12	0.76	0.2	0.74	0.23
difference	baseline + audio	0.84	0.11	0.78	0.18	0.86	0.12
difference	baseline + duration + language	0.86	0.1	0.80	0.16	0.78	0.21
difference	baseline + duration + language + audio	0.87	0.09	0.82	0.13	**0.87**	0.13
perpetual	baseline	0.61	0.12	0.48	0.09	0.59	0.26
perpetual	baseline + language	0.68	0.09	0.54	0.08	0.59	0.20
perpetual	baseline + duration	0.77	0.09	0.69	0.14	0.70	0.27
perpetual	baseline + audio	0.83	0.13	0.78	0.16	0.76	0.20
perpetual	baseline + duration + language	0.85	0.11	0.80	0.14	0.74	0.21
perpetual	baseline + duration + language + audio	0.86	0.09	0.80	0.12	0.82	0.14
wavelet	baseline	0.81	0.13	0.73	0.17	0.75	0.18
wavelet	baseline + language	0.74	0.11	0.66	0.10	0.69	0.16
wavelet	baseline + duration	0.84	0.12	0.78	0.18	0.74	0.27
wavelet	baseline + audio	0.86	0.11	0.80	0.14	0.83	0.13
wavelet	baseline + duration + language	0.85	0.11	0.80	0.14	0.79	0.21
wavelet	baseline + duration + language + audio	0.85	0.09	0.80	0.12	0.85	0.11
all	baseline	0.78	0.09	0.71	0.12	0.72	0.16
all	baseline + language	0.74	0.12	0.66	0.13	0.71	0.16
all	baseline + duration	0.83	0.07	0.75	0.14	0.79	0.21
all	baseline + audio	0.85	0.06	0.80	0.12	0.86	0.10
all	baseline + duration + language	0.84	0.11	0.78	0.16	0.80	0.17
all	baseline + duration + language + audio	**0.87**	0.09	**0.82**	0.15	0.85	0.11

Table 6: Performance metrics