Mod9 Alignment API
Version 0.8

The Mod9 Alignment API allows audio recordings to be synchronized with text transcriptions, producing timestamps for every word. In addition, a score is provided to quantify how well the given text matches speech in the audio, according to an acoustic model that is trained on standard American English, Latin American Spanish, European French, or German.

0. Endpoint and version

The current version of the API endpoint is **`v0.8`**

```
http://SERVER/mod9/align/v0.8
```

1. Basic usage

In RESTful style, resources are accessed via the HTTP GET verb. For example:

```
http://SERVER/mod9/align/v0.8?audio=AUDIOURL&text=TEXTURL
```

Note the two parameters to indicate publicly accessible URLs of **`audio`** and **`text`** files. The input audio should be WAV-formatted, or something easily understood by common tools such as ffmpeg or sox. The input text should be whitespace-delimited words, and should be encoded as UTF-8. The input is accepted even if it is encoded as invalid UTF-8; in this case any bogus characters are replaced with the special Unicode replacement character `U+FFFD`. So if you see that weird symbol in the alignment output, it probably indicates that your inputs were messed up.

The API service should return the alignment as a JSON structure. For example:

```
{"score": -67.83, "alignment": [["cost", 0.67], ["of", 1.04],
["preparing", 1.13], ["one", 1.62], ["hundred", 1.77], ["time",
2.02], ["cards", 2.32], ["will", 2.65], ["take", 2.79], ["an",
2.94], ["estimated", 3.02], ["1167", 3.80]]}
```

The numbers paired with each word in the alignment correspond to start times. The score should indicate how well the transcription matches the acoustics: higher is better (but note the negative sign).

The alignment service should report any errors (aside from network or HTTP errors) as JSON-formatted messages.

2. Alternate usage

You can also access the API via a POST request, such as:

```
POST /align/v0.8 HTTP/1.0
Accept: application/json
Content-Length: 43
Content-Type: application/json
Host: api.mod9.com

{"audio":"blah","words":["this","is","it"]}
```

The body of the message is a JSON-formatted structure. As with the GET request, the **`"audio"`** key corresponds to an accessible URL. However, the **`"words"`** key is a list of strings. Be sure to include an HTTP `Content-Length` header. Note that this POST is different from a standard Web browser's requests. The `Content-Type` is `application/json` rather than, say, `application/x-www-form-urlencoded`.

3. Configurable parameters

The alignment API accepts the following parameters:

`audio`: URL of audio file to be downloaded
`text`: URL of transcript text file to be downloaded (for GET requests)
`words`: JSON-formatted list of words in transcript (for POST requests)
`mode`: select a processing mode. Possible values:
- **`respond`**: (default) return a single response when finished
- **`stream`**: continuously return status messages, until finished
- **`submit`**: immediately return a job ID and process ID
- **`poll`**: given a jobid, return a single status message
- **`cancel`**: given a jobid, kill the process and update the status

`jobid`: required for **`mode=poll`**
`prune`: an integer value (default: 0) to control the algorithmic complexity
`skip`: `"True"` or `"False"` (default: `"False"`) to control the quality of alignment
`endtimes`: `"True"` or `"False"` (default: `"False"`) include word end times
`wordconf`: `"True"` or `"False"` (default: `"False"`) return per-word likelihoods
`lang`: spoken language to be used for modeling (default: `"en"`, i.e. U.S. English)
`metadata:` JSON-formatted object, for Mod9's reference

4. Alignment speed and quality

A couple parameters affect the resulting alignments:

`prune`: takes an integer value. By default is is set to '0', meaning that the alignment process will do an exhaustive search and will be guaranteed to return a result. This might be slow because it is a quadratic-time algorithm. If `prune` is set appropriately, the complexity becomes approximately linear-time, and a result may be returned in

much less time. However, if `prune` is too low, you may get a garbage result – or the alignment may fail, with a reported error. The trick is to pick a "reasonable" value, which depends on the quality of the audio and how well the transcription matches. Alternatively, the pruning value may a negative number, which means that it this is an increment applied each time the algorithm fails due to over-pruning. For example, `prune=-500` will first try a prune threshold of 500, then 1000, then 1500, etc... it will keep iterating through these increasing thresholds until it eventually succeeds. Use this if you don't want to deal with handling the pruning thresholds yourself and don't mind the processing taking a long time if you set the initial threshold too low. Note that the status progress messages will reset the percentage counter each time the alignment re-starts after encountering over-pruning.

`skip`: takes the values `"True"` or `"False"` (case-sensitive, as in Python). The default is `"False"`, meaning that the alignment process will try to match every single word in the transcription. Setting this to `"True"` means that some words may be skipped over if the match is not good. This allows results to be produced at lower pruning thresholds, but it may also result in less accurate alignment. This option might also be useful for cases where the transcription has a lot of words that aren't actually spoken (e.g. formatting symbols that weren't properly removed).

`endtimes`: takes the values `"True"` or `"False"` (case-sensitive, as in Python). The default is `"False",` meaning that only the word start time is returned. Otherwise, a pair of times is returned, additionally providing the end time for each word.

`wordconf`: takes the values `"True"` or `"False"` (case-sensitive, as in Python). The default is `"False",` meaning that an alignment comprises word-time pairs. Otherwise, a triple of word-time-score is returned, additionally providing a measure of alignment "confidence" for each word. This is really a per-frame log likelihood, and may or may not be interpretable as you'd expect...

`lang`: takes the values `"en"`, `"es"`, `"fr"`, or `"de"` (standard case-sensitive language codes). The default is `"en",` meaning that an alignment will be done using models trained on U.S. English. The possibilities will be updated in the future. One thing to keep in mind is that these acoustic models use *graphemic* pronunciation units, which may be inferior to *phonemic* units. Consider reverting to v0.7 for English if you prefer the prior alignment results, based on phonemic models.

5. Example of monitoring and canceling job execution

Here's how to use the polling mode with job IDs, to monitor and cancel an active job:

(1) Submit a job and note its job ID:
REQUEST:
`http://YOURSERVER/mod9/align/v0.8?audio=YOURAUDIOURL&text=YOURTEXTURL&mode=submit`
RESPONSE:
```
{"pid": 1234, "jobid": "YOURJOBID"}
```

(2) Poll the job ID, and note its status
REQUEST:
`http://YOURSERVER/mod9/align/v0.8?mode=poll&jobid=YOURJOBID`
RESPONSE:
```
{"status": "aligning 13566/42796 (31.7 %)", "jobid": "YOURJOBID"}
```

(3) Cancel the job, and note its updated status
REQUEST:
`http://YOURSERVER/mod9/align/v0.8?mode=cancel&jobid=YOURJOBID`
RESPONSE:
```
{"status": "cancelled", "jobid": "YOURJOBID"}
```

(4) Poll its status again
REQUEST:
`http://YOURSERVER/mod9/align/v0.8?mode=poll&jobid=YOURJOBID`
RESPONSE:
```
{"status": "cancelled", "jobid": "YOURJOBID"}
```

Note that you should first try to cancel a job using this API, rather than trying to kill the process ID by accessing the alignment server facility such as:

`http://YOURSERVER/mod9/stat/kill?p=1234`

Use this latter mechanism only as a last resort, as it may interfere with the processing of other requests that could be polling a given job. Please refer to the Mod9 Alignment Server documentation for more details on server management.

6. Metadata

By default, data sent to the API service will be transferred to Mod9, and will be used for development purposes. Your data may be used to train statistical models for speech recognition – which consequently could be specifically targeted to your application domain. In order for this transcribed speech data to be used most effectively, it is helpful to provide some additional metadata so that Mod9 can organize and process large training sets from its customers.

Such information can be sent in API requests using the **`metadata`** parameter:

- **GET** requests: the value should be a **URL-safe base64** encoding of a JSON object.

- **POST** requests: include a **field called `"metadata"`** in the JSON object that constitutes the request body.

The exact contents of the JSON object are not specified. You are encouraged to submit any fields of information that could be potentially helpful to characterize the data. Please submit metadata in a format that is as informative (and consistent) as possible – especially if it describes the speaker, language, and acoustic environment.

Depending on your API terms of use, it may be possible to send a `"mod9_exclude"` parameter in the metadata object. This indicator will be understood to mean that Mod9 should not transfer and retain data for certain requests (e.g. containing specifically licensed content that cannot be shared). Note that such data will temporarily remain on the alignment server, for the purpose of debugging; it may be possible to clean out these caches using a function of the Mod9 Alignment Server:

```
http://YOURSERVER/mod9/clean/delete
```

7. Example: sending metadata in Python

Here is a demonstration of GET and POST requests that pass along metadata.

```
# Dependencies
import urllib2
import urllib
import base64
import json

# Be sure to use the right version
endpoint = 'http://YOURSERVER/mod9/align/v0.5'

# Data to be submitted
audiourl = 'http://arlofaria.com/test.wav'
texturl = 'http://arlofaria.com/test.txt'
metadata = {'title':'A test title', 'author':'Arlo Faria'}

# GET request
metadata_encoded = base64.urlsafe_b64encode(json.dumps(metadata))
params_encoded = urllib.urlencode({'audio':audiourl,
                                   'text':texturl,
                                   'metadata':metadata_encoded})
url = endpoint + '?' + params_encoded
print urllib2.urlopen(url).read()

# POST request: encode 'words' as list, rather than 'text' as URL
words = urllib2.urlopen(texturl).read().split()
body = json.dumps({'audio':audiourl,
                   'words':words,
                   'metadata':metadata})
req = urllib2.Request(endpoint, data=body,
                      headers={'Content-type':'application/json'})
print urllib2.urlopen(req).read()
```

8. Logging

If you have access to a Mod9 Alignment Server, logs may be viewed at:

`http://YOURSERVER/mod9/logs/align`
- Optional argument: `date=YYYY-MM-DD` (defaults to today)
- This will report the amount of data processed by the server

Please contact Mod9 if a problem cannot be resolved by inspecting these logs.

9. Troubleshooting

Generally, problems with the Mod9 Alignment API service are related to either the speed of processing very long inputs, or the quality of alignments for certain types of input. These are difficult problems that Mod9 is actively trying to resolve.

In general, for best results no pruning should be applied. In practice, applying a pruning threshold may significantly speed up the processing time for long inputs (e.g. over a minute in duration). However, just because a processing run completes – even if it's with no pruning – the results could still be garbage. That's because of a number of factors, including possibly:

1. crappy audio quality: listen to the audio and make sure it sounds OK
2. crappy transcripts: see if it's actually matching what was said
3. crappy vocabulary: the pronunciation or rare and unusual words may be wrong
4. crappy acoustic models: could be retrained to better match your data
5. crappy language: it needs to be English, preferably a standard American dialect
6. crappy non-speech modeling: especially if there are sounds effects or music, it won't match the standard acoustic model that expects low-energy silence
7. crappy algorithm: this Viterbi decoding was designed for aligning utterances of several seconds in length. For much longer inputs, a different approach is needed.

The first two items are mostly in the user's control (garbage in, garbage out), whereas the other items are research and development for Mod9 to address.

A note about timing: the times reported by the server will be formatted as `"%.2f"`. In reality, however, the underlying signal processing operates on discrete-time centisecond frames corresponding to 25ms analysis windows – without padding or reflection. So if you need truly centered times, add 0.0125s to each time; also, don't expect that the last time corresponds to the duration of the audio, which may be up to 0.0375s longer.

A note about non-words in the transcripts... due to some text pre-processing, some symbols in the transcripts may be treated as non-words – for example, the HTML tag `<br>`. If the transcript is empty or has only non-words, the alignment will be emtpy. Note that some word confidence values may be `null` – for weird reasons.

10. Contact

This API service was designed and implemented by Arlo Faria:

api@mod9.com
+1 (510) 705-3223