MLMI14: Keyword Spotting

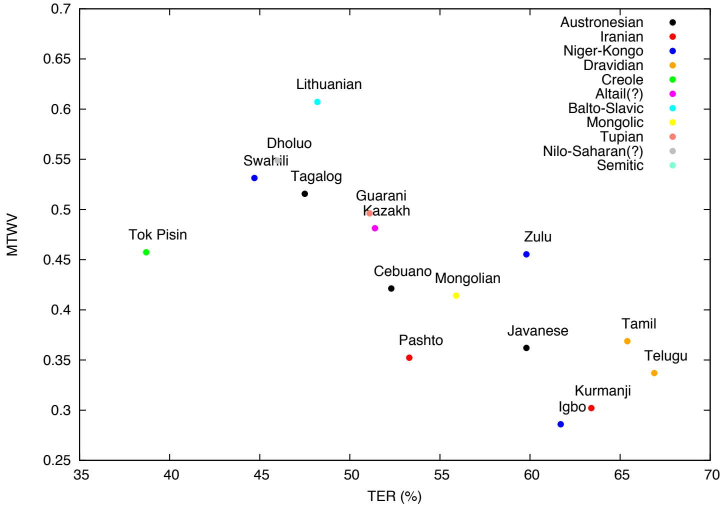
Mark Gales

Lent 2022

Babel: Low-Resource ASR and Keyword-Spotting (IR)

"The Babel Program will develop agile and robust speech recognition technology that can be rapidly applied to any human language in order to provide effective search capability for analysts to efficiently process massive amounts of real-world recorded speech." - Babel Program BAA

Highly Diverse Languages - ASR/KWS Performance

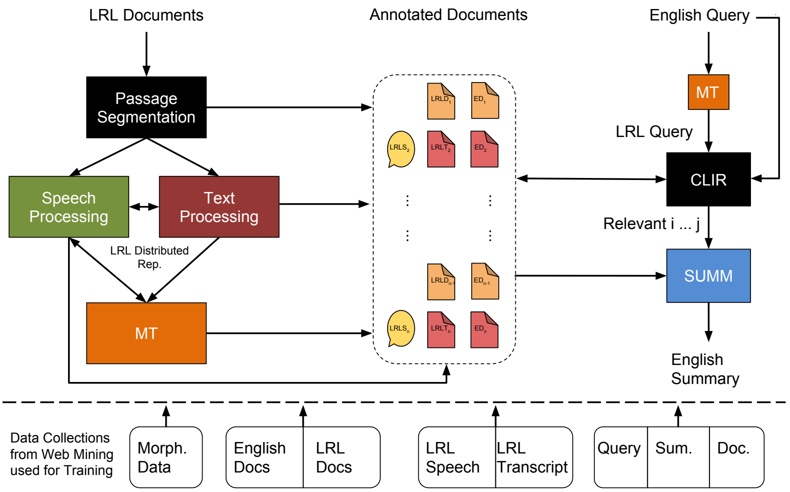


MATERIAL: Cross-Language Information Retrieval

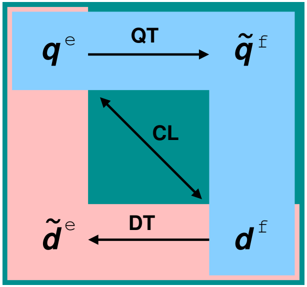
MATERIAL

“ ... - MATERIAL - that will enable users to quickly develop and deploy fully automatic systems that will allow English-only speakers to accurately and efficiently identify foreign language documents of interest across social media, newswire, and broadcasts - to name a few.” - IARPA press release

Cross-Language Information Retrieval and Summarisation



Cross-Language Information Retrieval (CLIR)

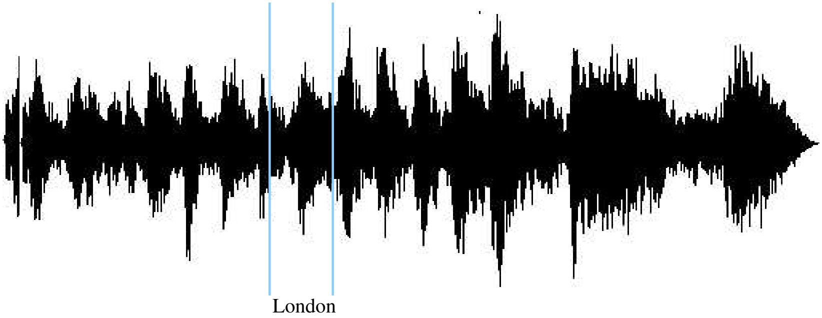


- $\boldsymbol{q}^{e}$: English query
- $\tilde{\boldsymbol{q}}^{f}$: translated English query
- $\boldsymbol{d}^{f}$: foreign document
- $\tilde{\boldsymbol{d}}^{e}$: translated foreign document

- Standard CLIR approaches include
  - query translation (QT) - ($\boldsymbol{q}^{e} \rightarrow \tilde{\boldsymbol{q}}^{f}$) search $\boldsymbol{d}^{f}$
  - document translation (DT) - $\boldsymbol{q}^{e}$ search ($\boldsymbol{d}^{f} \rightarrow \tilde{\boldsymbol{d}}^{e}$)
  - cross-lingual embedding matching (CL) - $\boldsymbol{q}^{e}$ search $\boldsymbol{d}^{f}$

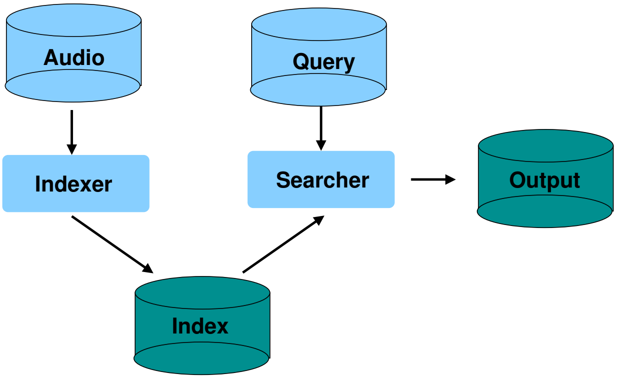
Keyword Spotting

Keyword Spotting Task



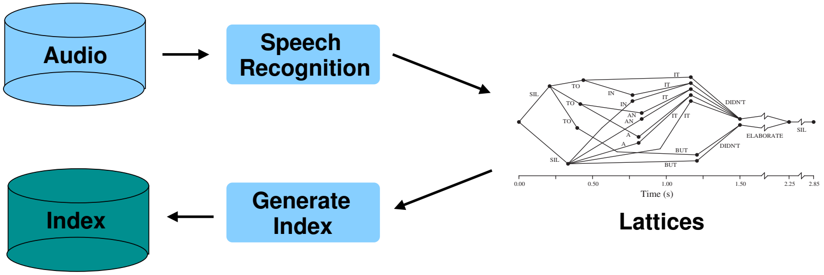
- Detect key word (or phrase), query in the audio
  - require location (utterance/time) information in audio
  - when multiple words sometimes called Spoken Term Detection
- Query may be written (focus here) or spoken

General Process



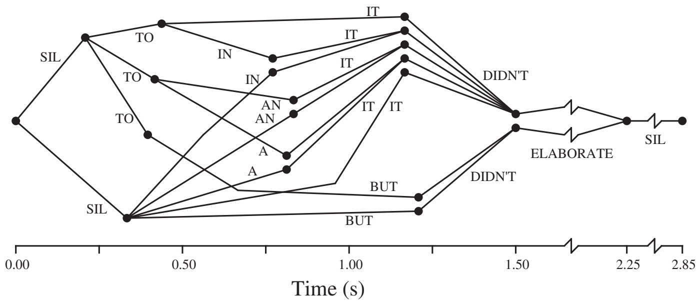
- Often split into two distinct parts for efficiency
  - indexer - can be slow, supplies information for ...
  - searcher - needs to be fast, and handle any query

Index Generation



- Systems make use of ASR systems
  - generate lattices, convert to index
- Contrast to ASR output, need highly rich lattices
  - interested in more than 1-best output

Lattices



- A lattice, $\mathcal{L}$, comprises:
  - nodes (sometimes called state): associated with time stamps
  - arcs: have labels and scores (not shown)
- The labels for the lattice above are words
  - sub-words (phones) may also be used

Word Search Strategy

- Initially just consider detecting whether a label, $\tilde{w}$, occurs
  1. retrieve all arcs, a, in the index with required label $\tilde{w}$, $\mathcal{I}(\tilde{w})$
  2. compute the posterior for that arc in the lattice $P(a|\mathcal{L}(a))$
  3. construct a count for $\mathcal{C}(\tilde{w}|\mathcal{L})$

$$\mathcal{C}(\tilde{w}|\mathcal{L}) = \sum_{a \in \mathcal{I}(\tilde{w}):\mathcal{L}(a)=\mathcal{L}} P(a|\mathcal{L}(a))$$

  4. define a threshold of $\mathcal{C}(\tilde{w}|\mathcal{L})$ for existence of word in utterance

- Yields a count for a particular label for a lattice.
  - the only question is how to obtain the posterior efficiently ...

Index Content

- Indexer needs to extract information from the lattice
    - need to support rapid calculation of arc posteriors
- Basic information extracted for each arc a with label w:
    - lattice identity: $\mathcal{L}(a)$
    - input node: $k(a)$
    - output node: $n(a)$
    - score: $P(a|k(a))$
    - probability mass leading to node from initial node(s): $\alpha(k(a))$
    - probability mass from output node to final node(s): $\beta(n(a))$
- An index is generated for each label w, $\mathcal{I}(w)$

Arc Posterior Calculation

- Need to compute the arc posterior, $P(a|\mathcal{L}(a))$
  - standard automata problem (not discussed in this course)
  - simple to express as (π is a path in the lattice)

$$P(a|\mathcal{L}(a)) = \frac{\sum_{\pi\in\mathcal{L}(a):a\in\pi} P(\pi|\mathcal{L}(a))}{\sum_{\pi\in\mathcal{L}(a)} P(\pi|\mathcal{L}(a))}$$

- Choice of options, simplest consider (see HMM training):
  - $\alpha(k(a))$: the forward-probability to the input node for arc a
  - $\beta(n(a))$: the backward-probability to the output node for arc a
- The numerator can then be written as

$$\sum_{\pi\in\mathcal{L}(a):a\in\pi} P(\pi|\mathcal{L}(a)) = \alpha(k(a))P(a|k(a))\beta(n(a))$$

Phrase Search Strategy

- A similar scheme for phrases (or sub-words) can be used,
  - but now consider multiple arcs
  - the count above now becomes for $\tilde{\boldsymbol{w}} = \tilde{w}_1, \ldots, \tilde{w}_n$

$$\mathcal{C}(\tilde{\boldsymbol{w}}|\mathcal{L}) = \sum_{\pi \in \mathcal{L}, \tilde{\boldsymbol{a}} \in \pi} P(\boldsymbol{a}|\mathcal{L})$$

  - $\boldsymbol{a} = a_1, \ldots, a_n,\ a_i \in \mathcal{I}(\tilde{w}_i),\ k(a_{i+1}) = n(a_i)$
- Still need the posterior, but easy to get as

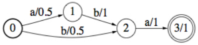
$$\sum_{\pi \in \mathcal{L}(a_1) : \boldsymbol{a} \in \pi} P(\pi|\mathcal{L}(a)) = \alpha(k(a_1)) \left(\prod_{i=1}^{n} P(a_i|k(a_i)) \right) \beta(n(a_n))$$

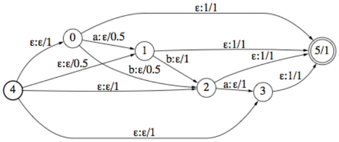
- Process often implemented as a WFST for flexibility ...

WFST Index Implementation

- Index generation process:
  1. normalise lattices by weight pushing
     - sum of paths from all arcs equals 1 (denominator=1)
  2. add two new nodes (start and end)
  3. for each original node add two arcs:
     - from start node to node with forward-probability to node
     - from node to end node with backward-probability to node

- Simple example (no optimisation of final WFST)





Handling OOV Words

- Assumed that the query term $\tilde{\boldsymbol{w}}$ is in-vocabulary
  - if any query term is not in the vocabulary it cannot be found
  - will not occur in any of the utterances
- The query terms are often not known in advance

How to handle out-of-vocabulary (OOV) query terms?

- There are a number of approaches which will be discussed:
  1. map word lattices to phones (and apply phone confusions)
  2. map OOV terms into IV terms proxy keywords
  3. generate lattices with minimal (or no) OOV terms

Phone Confusions

- Consider:
  - query: `ABANDON` - out of vocabulary word
  - `ABANDONED` is in-vocabulary

- Map all query and arc labels to phones

`ABANDON`	ax b ae n d ax n
`ABANDONED`	ax b ae n d ax n d

  - requires query phone sequence to be present in lattice
  - basically ignores word boundary information

- To increase chance of finding a hit introduce phone confusion
  - for example $P(/\mathtt{a}/|/\mathtt{b}/)$ is used to expand lattice/query
  - dramatically increases the number of hits found
  - can include phone confusion score as part of count, $\mathcal{C}(\tilde{w}|\mathcal{L})$

Proxy Keywords

- Aim is to get closest (set of) IV word to the OOV query
  - again use phone sequence consider query `ABANDON`

`ABANDON`	ax b ae n d ax n

Decoding Vocabulary

`AARDVARK`	aa d v aa k
	⋮
`ABANDONED`	ax b ae n d ax n d
	⋮

  - find "closest" IV word to query

$$P(\text{ABANDONED}|\text{ABANDON}) =$$
$$P(/\texttt{ax}/|/\texttt{ax}/)P(/\texttt{b}/|/\texttt{b}/)P(/\texttt{ae}/|/\texttt{ae}/)\ldots P(/\texttt{n}/|/\texttt{n}/)P(/\texttt{d}/|\epsilon)$$

- Converts all OOV terms to closest IV terms - IV search
  - can include "distance" to IV word in count, $\mathcal{C}(\tilde{w}|\mathcal{L})$

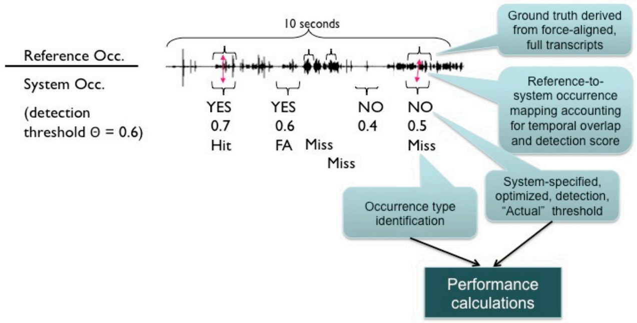
Morphological Decomposition

- Apply morphological decomposition on vocabulary and query

ABANDON	ABANDON
ABANDONED	ABANDON+ +ED

- Possible to apply decomposition to word (label) arcs
  - limited coverage of OOV terms (consider word lattices)
- Alternatively run alternative decoding
  1. morphological decomposition on LM training data - train LM
  2. decoding using morph LM (often without changing AM)
  3. decompose query terms and perform search
- Automated approaches (based on character strings) also used
  - morphessor, byte pair encoding

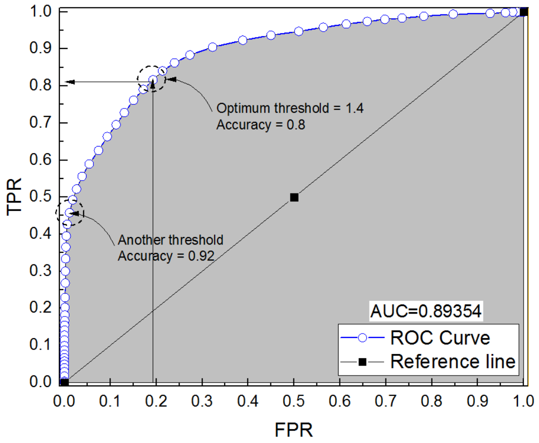
Assessment - Term-Weighted Value (TWV)



10 seconds
Reference Occ.
System Occ.
(detection threshold Θ = 0.6)
YES 0.7 Hit
YES 0.6 FA
Miss
Miss
NO 0.4
NO 0.5 Miss
Ground truth derived from force-aligned, full transcripts
Reference-to-system occurrence mapping accounting for temporal overlap and detection score
System-specified, optimized, detection, "Actual" threshold
Occurrence type identification
Performance calculations

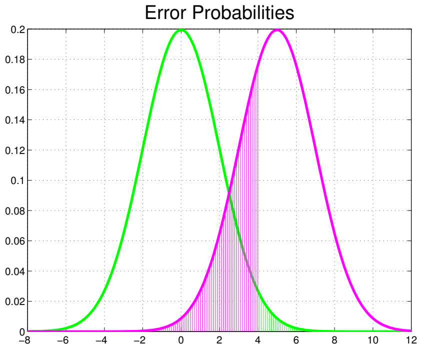
- For each query term $\tilde{\boldsymbol{w}}$ at threshold θ - for Babel $\beta = 999.9$
  - compute over all utterances:
    $\text{TWV}(\tilde{\boldsymbol{w}}, \theta) = 1 - [P_{Miss}(\tilde{\boldsymbol{w}}, \theta) + \beta P_{FA}(\tilde{\boldsymbol{w}}, \theta)]$
- Predict threshold for average score - Actual TWV (ATWV)
  - or select best threshold on test set - Maximum TWV (MTWV)

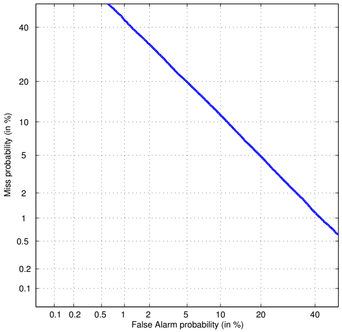
Assessment - Area Under Curve



- Overall performance of system (no threshold selected)
    - rank-order based, not impacted by (monotonic) calibration

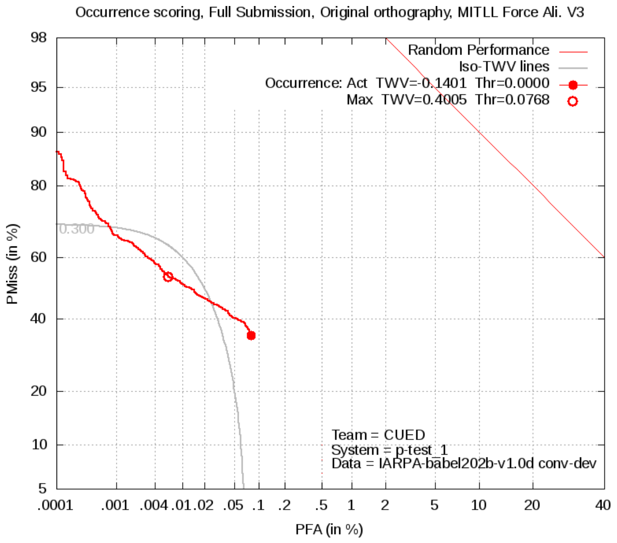
Detection Error Trade-Off Curves





- Results often plotted on DET curves rather than ROC curves
  - avoids pushing curve too far into the top-left corner
  - Gaussian distributed distributions mapped to straight lines
- Also used for speaker verification

Example Performance: Swahili



Refinements and Extensions

- Timing Information: if the location of the query in an audio stream is required then timing information must be added to the index. Start and end times are added (as well as clustering time stamps together).
- Score Normalisation: rather than taking the raw posteriors, the scores are normalised for each keyword. The simplest form of this is sum-to-one normalisation.
- System Combination: in the same fashion as ASR (and many other tasks) combining multiple KWS systems together has become increasingly popular. Here multiple indexes are generated, for example word and morph systems, and the outputs combined together.
- Some of these approaches will be examined in the practical for this module.

Example KWS performance - Zulu

- Limited resources can yield high OOV rates
  - significant problem for agglutinative languages
  - Zulu Limited Language Pack: 61% dev query terms OOV

KWS Process	MTWV		
	IV	OOV	Tot
Word	0.2655	0.0000	0.1033
+phone	0.2596	0.0970	0.1606
+cascade	0.2609	0.0970	0.1611
+lm0	0.2649	0.1338	0.1851
+morph	0.2615	0.2073	0.2287

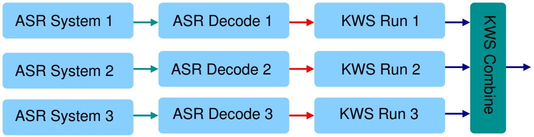
- Range of approaches developed to address OOVs
  - +phone: map lattice to phones, phone KWS (with confusions)
  - +cascade: treat missed IV terms as OOV
  - +lm0: set LM scores to zero for OOV search
  - +morph: generate morph lattices, do IV search for morphs

System Combination for KWS

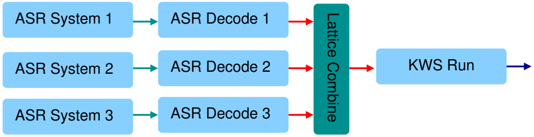
- System combination extensively used during Babel evaluation
  - ASR system combination over IBM, RWTH, CUED
  - KWS system combination over IBM, RWTH, CUED
- Posting-list merging is a standard approach for KWS
  - see practical for details ...
- Significant gains possible - for Javanese (Full Language Pack)

BottleNeck Features	TER (%)	MTWV IV	MTWV OOV	MTWV TOT
Aachen (A28)	52.5	0.4787	0.4379	0.4736
IBM (I28)	52.1	0.4763	0.4283	0.4712
A28$\otimes$I28	50.9	0.4979	0.4843	0.4970

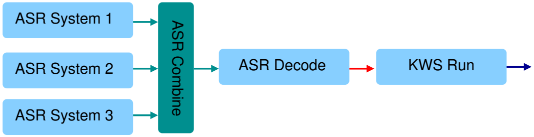
KWS System Combination Architectures



Posting-List Combination (See Practical)



Lattice Combination

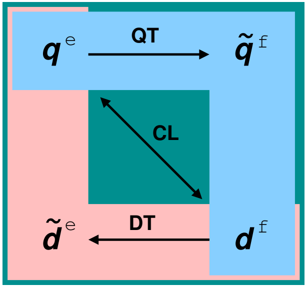


ASR System Combination

Cross-Language Information Retrieval (Reference)

Cross-Language Retrieval Approaches

- Consider English query ($\boldsymbol{q}^{\mathtt{e}}$) and foreign document ($\boldsymbol{d}^{\mathtt{f}}$)



$\boldsymbol{q}^{\mathtt{e}}$
QT
$\tilde{\boldsymbol{q}}^{\mathtt{f}}$
CL
$\tilde{\boldsymbol{d}}^{\mathtt{e}}$
DT
$\boldsymbol{d}^{\mathtt{f}}$

- Standard CLIR approaches include
  - query translation (QT) - ($\boldsymbol{q}^{\mathtt{e}} \rightarrow \tilde{\boldsymbol{q}}^{\mathtt{f}}$) search $\boldsymbol{d}^{\mathtt{f}}$
  - document translation (DT) - $\boldsymbol{q}^{\mathtt{e}}$ search ($\boldsymbol{d}^{\mathtt{f}} \rightarrow \tilde{\boldsymbol{d}}^{\mathtt{e}}$)
  - cross-lingual embedding matching (CL) - $\boldsymbol{q}^{\mathtt{e}}$ search $\boldsymbol{d}^{\mathtt{f}}$

Example English Queries

Type	Description	Example
Simple	word	bribery
	phrase	`"military force"`
	conjunction	`"military force",attack`
Constrained	morphological	`<dogs>`
	hypernyme	`nurse[hyp:profession]`
	synonyme	`retreat[syn:withdraw]`
	event frame	`virus[evf:medical]`
Conceptual	full	`"freshwater fish"+`
	example of	`EXAMPLE_OF(freshwater fish)`
Hybrid	any combination	`zoo+,environment`

- Very rich query specification language
  - designed specifically for MATERIAL – no prior work/data

Simple Query Processing

Input	Output ($\boldsymbol{q}^{\mathrm{e}}$)
`"military force"`	{{military:1.0},{force:1.0}}
`<dogs>`	{{dogs:1.0}}
`nurse[hyp:profession]`	{{nurse:1.0}}
`"freshwater fish"+`	{{freshwater:1.0},{fish:1.0}}
`EXAMPLE_OF(freshwater fish)`	{{freshwater:1.0},{fish:1.0}}

Table: Example CUED query processing

- Many queries are not simple words – need to handle
  - phrases, semantic and conceptual expansions
- There are multiple ways to handle phrases
  - word conjunctions (all teams), Markov models (early CLIR)
- Semantic and conceptual expansion types
  - numerous word2vec-style options (SCRIPTS), none (CUED)

Example QT: Query Generation Model

- Probability of generating query $\boldsymbol{q}^{\mathrm{e}}$ from document $\boldsymbol{d}^{\mathrm{f}}$

$$P(\boldsymbol{q}^{\mathrm{e}}|\boldsymbol{d}^{\mathrm{f}}) = \prod_{w^{\mathrm{e}} \in \boldsymbol{q}^{\mathrm{e}}} \left[(1-\alpha) P(w^{\mathrm{e}}|\boldsymbol{d}^{\mathrm{f}}) + \alpha P(w^{\mathrm{e}}|\boldsymbol{g}^{\mathrm{e}}) \right]$$

  - $P(w^{\mathrm{e}}|\boldsymbol{d}^{\mathrm{f}})$, $P(w^{\mathrm{e}}|\boldsymbol{g}^{\mathrm{e}})$ query term generation probabilities
- Foreign language document based

$$P(w^{\mathrm{e}}|\boldsymbol{d}^{\mathrm{f}}) = \sum_{w^{\mathrm{f}} \in \boldsymbol{d}^{\mathrm{f}}} P(w^{\mathrm{e}}|w^{\mathrm{f}}) P(w^{\mathrm{f}}|\boldsymbol{d}^{\mathrm{f}})^{\epsilon}$$

  - $P(w^{\mathrm{e}}|w^{\mathrm{f}})$ translation probability table, ϵ soft "occurrence"
- General English document, $\boldsymbol{g}^{\mathrm{e}}$, based (background model)

$$P(w^{\mathrm{e}}|\boldsymbol{g}^{\mathrm{e}}) = \sum_{w \in \boldsymbol{g}^{\mathrm{e}}} \delta(w^{\mathrm{e}}, w) P(w|\boldsymbol{g}^{\mathrm{e}})$$

  - $\delta(w^{\mathrm{e}}, w)$ indicator function (1 if $w^{\mathrm{e}} = w$, 0 otherwise)

CLIR for Speech

- Consider speech-only acoustic "documents", $\boldsymbol{x}^{\mathrm{f}}$
  - use ASR to convert into lattice/CN $\boldsymbol{x}^{\mathrm{f}} \rightarrow \mathcal{L}^{\mathrm{f}}$
  - each arc, a, of the lattice, $a \in \mathcal{L}^{\mathrm{f}}$ contains:
    label (foreign word) $\mathtt{lab}(a)$; score $\mathtt{score}(a) = P(a|\mathcal{L}^{\mathrm{f}})$
- Options available
  - 1-best ASR hypotheses (contain scores)
  - lattices or confusion networks (contain alternatives and scores)
- Use $P(w^{\mathrm{f}}|\mathcal{L}^{\mathrm{f}})$ to support alternatives/scores
  - query generation model includes $P(w^{\mathrm{f}}|\boldsymbol{d}^{(f)}) = P(w^{\mathrm{f}}|\boldsymbol{x}^{\mathrm{f}})$

$$P(w^{\mathrm{e}}|\boldsymbol{x}^{\mathrm{f}}) = \sum_{w^{\mathrm{f}} \in \mathcal{L}^{\mathrm{f}}} P(w^{\mathrm{e}}|w^{\mathrm{f}}) P(w^{\mathrm{f}}|\mathcal{L}^{\mathrm{f}})^{\epsilon}$$

Speech "Document" Term Probabilities

- Computing Word probabilities from lattices same as KW
- Use lattice/CN posteriors $P(w^{\tt f}|\mathcal{L}^{\tt f})$ to estimate
  - probability of $w^{\tt f}$ in $\mathcal{L}^{\tt f}$

$$P(w^{\tt f}|\mathcal{L}^{\tt f}) = \frac{\sum_{a\in\mathcal{L}^{\tt f}} \delta(w^{\tt f}, {\tt lab}(a))P(a|\mathcal{L}^{\tt f})}{\sum_{a\in\mathcal{L}^{\tt f}} P(a|\mathcal{L}^{\tt f})}$$

- For efficiency standard WFST KWS-style operations are used

$$\begin{array}{ccccccc} \boldsymbol{x}^{\tt f} & \rightarrow & \mathcal{L}^{\tt f} & \rightarrow & \tilde{\mathcal{L}}^{\tt f} & \rightarrow & \mathcal{I}^{\tt f} \\ & \text{ASR} & & \text{Push} & & \text{Det/Ind} & \end{array}$$

  - for simplicity notation just uses $\mathcal{L}^{\tt f}$

Speech Index Implementation

- Each word (lattice arc) is treated separately in current model
  - effectively a *bag-of-word* models
  - multiple lattices are generated for each acoustic-document
- For each acoustic-document $\boldsymbol{x}^{\mathtt{f}}$ there are a sequence of lattices

$$\mathcal{L}^{\mathtt{f}} = \left\{\mathcal{L}^{\mathtt{f}}_1, \ldots, \mathcal{L}^{\mathtt{f}}_N\right\}$$

  - the extension of the probabilities becomes, for example

$$P(w^{\mathtt{f}}|\mathcal{L}^{\mathtt{f}}) = \sum_{\mathcal{L}\in\mathcal{L}^{\mathtt{f}}} \sum_{a\in\mathcal{L}} \delta(w^{\mathtt{f}}, \mathtt{lab}(a)) P(a|\mathcal{L}) \Big/ \sum_{\mathcal{L}\in\mathcal{L}^{\mathtt{f}}} \sum_{a\in\mathcal{L}} P(a|\mathcal{L})$$

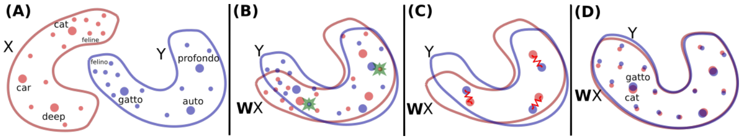
  - for search efficiency use a word-level index $\forall w^{\mathtt{f}}, \forall \boldsymbol{x}^{\mathtt{f}}$

$$\mathtt{idx}(w^{\mathtt{f}}|\boldsymbol{x}^{\mathtt{f}}) = P(w^{\mathtt{f}}|\mathcal{L}^{\mathtt{f}})$$

  - this is stored and allows rapid query search

Example CL: Multi-Lingual Embeddings

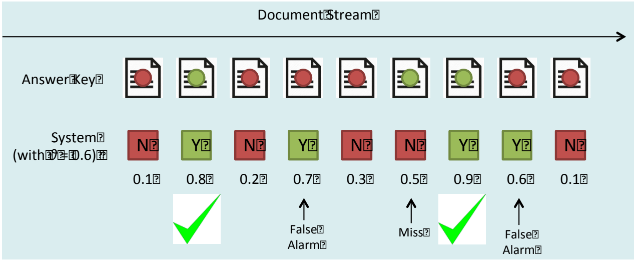
- Often limited quantities of bitext (machine translation) data
  - build mono-lingual embedding spaces for languages of interest
  - map embeddings to the same space using limited bitext data



Steps: **(B)** rotation; **(C)** Procrustes trans.; **(D)** Refinement

  - map query $\boldsymbol{q}^{\mathtt{f}}$ and document $\boldsymbol{d}^{\mathtt{e}}$ into multi-lingual space
- Convert "distances" in multi-lingual space into probabilities

Assessment - Query Weighted Value (QWV)



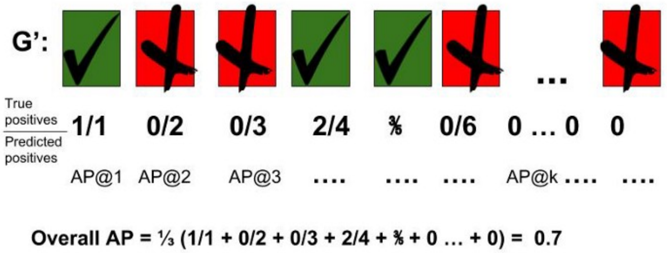
- Similar to term weighted value for KWS
- Simple average of weighted retrieval errors

$$\text{QWV}(\boldsymbol{q}^{\text{e}}, \theta) = 1 - P_{\texttt{miss}}(\boldsymbol{q}^{\text{e}}, \theta) + \beta P_{\texttt{fa}}(\boldsymbol{q}^{\text{e}}, \theta)$$

  - β controls the operating point, ϑ threshold

Assessment - Mean Average Precision (mAP)

- Using IR system rank order all documents for each query
  - compute precision as document recall changes for each query
  - average the average precision over all queries



- Simple example above for a particular query
  - documents containing query ranked: 1, 4, 6
  - mean average precision for query: 0.7

Performance (Query Translation)

Language	ASR System	WER@ANA		mAP	
		CTS	WB	1-Best	Lat
Swahili	CUED1	36.0	31.5	0.2058	0.2088
Bulgarian	CUED1	32.6	18.9	0.7366	0.7413
Lithuanian	CUED1	41.8	24.4	0.6466	0.7049
	CUED2	37.4	21.4	0.6666	0.7477
	CUED3	35.8	20.6	0.6948	0.7440

- Need to handle low-resource languages (similar to Babel)
- Speech "document" data sources:
  - conversational telephone speech (CTS)
  - podcasts/broadcast (WB) speech data
- Lattice search important for low-resource tasks