CWI at TREC 2012, KBA track and Session Track

Samur Araujo
Delft University of Technology,
PO Box 5031, 2600 GA
Delft, the Netherlands
s.f.cardosodearaujo@tudelft.nl

Gebrekirstos Gebremeskel
CWI
the Netherlands
gebre@cwi.nl

Jiyin He
CWI
the Netherlands
j.he@cwi.nl

Corrado Bosscarino
CWI
the Netherlands
corrado@cwi.nl

Arjen de Vries
CWI
the Netherlands
arjen.de.vries@cwi.nl

ABSTRACT

We participated in two tracks: Knowledge Base Acceleration (KBA) Track and Session Track. In the KBA track, we focused on experimenting with different approaches as it is the first time the track is launched. We experimented with supervised and unsupervised retrieval models. Our supervised approach models include language models and a string-learning system. Our unsupervised approaches include using: 1)DBpedia labels and 2) Google-Cross-Lingual Dictionary (GCLD). While the approach that uses GCLD targets the central and relvant bins, all the rest target the central bin. The GCLD and the string-learning system have outperformed the others in their respective targeted bins. The goal of the Session track submission is to evaluate whether and how a logic framework for representing user interactions with an IR system can be used for improving the approximation of the relevant term distribution that another system that is supposed to have access to the session information will then calculate.

the documents in the stream corpora. Three out of the seven runs used a Hadoop cluster provide by Sara.nl to process the stream corpora. The other 4 runs used a federated access to the same corpora distributed among 7 workstations.

1. INTRODUCTION

We have participated in two tracks this year: the KBA track and the session track. The paper is orgaized as follows. We specify our submission to the session track in Section 2. In Section 3 we will discuss our approaches, experiments and results with respect to the KBA track. We conclude the paper with our observations and findings in section 4.

2. SESSION TRACK

We consider a TREC search session to mainly provide evidence of a learning process. A task, as specified in the narrative, requires a user to interact with a search system for learning about the topic and about the modes of interaction with that particular system that can lead to task accomplishment.

The goal of this submission is to evaluate whether and how a logic framework for representing user interactions with an IR system can be used for improving the approximation of the relevant term distribution that another system that is supposed to have access to the session information will then calculate.

We distinguish two separate models. One is a retrieval model that ranks documents in response to the current query. Another model takes care of representing the effect of interactions to how the underlying model ought to rank documents. We use query expansion to connect the output of the interaction model to the input of the IR model: the interaction model provides therefore a set of expansion terms for the query that the user issued at the last step of the recorded session.

We use a variation of Probabilistic Dynamic Epistemic Logic (PDEL) [8] to represent user interactions as sentences interpreted on a Kripke structure. Within this approach, actions may change the model in more complex ways than state elimination only, hence the resulting probability update generally differs from Bayesian update. The underlying IR model is a classic language model implemented on Indri with standard parameters, emphasising the contribution of the interaction model to the overall performance.

The probabilities at the exit states of the model represent, for each agent, a subjective probabilistic statement: we hypothesize that a mixture of these probability metrics outperform current IR models with relevance feedback. We evaluated this hypothesis on the session track data.

We briefly introduce the model and some background literature. It is outside the scope of this document to work out all the technical details of this approach, which is only presented here as a motivation for the experiments.

2.1 Retrieval Model

Besides the most basic way to incorporate new evidence into an existing probabilistic model, that is conditional probability, there are some alternatives such as using Dempster-Shafer theory [5] or cross-entropy [4]. A model of a retrieval situation with PDEL contains two separate parts, one epistemic model that accomodates the deterministic information about the interactions and one pure probabilistic model.

A general case of such a mixed retrieval model is a relevance model as in [3] with a PDEL model built on top. Next to a frequentist or bayesian interpretation we introduce then an alternative interpreta-

tion of the relevance model in [3], that is the probability distribution over a vocabulary conditioned on the observation of a sample from the relevant population. Both the interpretations that Lavrenko puts forth in [3] assume a generative process that repeatedly samples terms from the relevant population. In the logical interpretation we extend the generative process by specifying that observations are performed in a certain order, with a well defined structure as determined by the protocol used to create the dataset.

According to this view, terms from clicked document can be distinguished from query terms and the end system can explore the model parameters θ more effectively in an attempt of estimating the relevant parameter setting θ_R. We still probabilistically evaluate an uncertainty about relevance, but now we take into account deterministic information such as the observed interactions.

2.1.1 Adding interactions to a relevance model

Assuming that probabilistic statements about individual cases (for example the probability of observing a certain term) depend on the knowledge state of an agent who estimates this probability [2], the main issue that we address in the experiments is: how can we modify existing IR models such as in [3] in order to accomodate a formalisation of these epistemic states as "observed" (with uncertainty modeled in PDEL) during a search session? We want to formalise the intuition that observing a term as a query, in a document from a result list or in a clicked document should result in different representations in the total model.

We make the simplifying assumption that we can keep the representation of documents and queries, the notion of relevance as well as the distribution over documents and queries as it is in the standard account [3]. We only assume that, next to query's and document's transforming functions $D(x)$ and $Q(x)$ for every x in the full representation space $\mathcal{S}$, there is a similar function $A(x)$ that transforms elements of $\mathcal{S}$ into well formed sentences of the PDEL language $\mathcal{L}_{PEL}$. This transforming function represents another possible reduction of the full representation space: next to documents and queries, a suitable subset of the space $\mathcal{S}$ represents actions that a user might perform during a search session. Since we still want the transforming functions, Q, D and now the freshly introduced A to be functionally similar to each other, we get the additional requirement that queries and documents must be regular sentences ϕ of the PDEL language as well. In this case the probability of observing an interaction expressed with the formula ϕ, becomes:

$$P(\phi) = \int_{\Theta} \prod_{a=1}^{n} \left(\sum_{x_{i_a}} \delta(d_a, x_{i_a}) P_{i_a,\theta}(x_{i_a}) \right) \prod_{b=n+1}^{m} \left(\sum_{x_{i_b}} P_{i_b,\theta}(x_{i_b}) \right) p(\theta)$$

which is the same as [3, eq.3.4], where the argument of the first product operator are the dimensions of the full representation space that are retained by A.

Under this assumption a modification of the generative process that produces an alternative distribution $P(\cdot|\mathbf{r})$ for the relevant population, provided that it outputs term probabilities in the usual way, still admits standard ranking approaches such as PRP or KL divergence based methods.

But, even if we make plausible that only the relevance model needs to be modified, what additional structures from [8] do we need? The task here is to do something similar to [3, eq.3.9], where an initial estimate $P(\theta)$ was updated with the observation that $\mathbf{r}$ by means of Bayes' rule, as

$$p(\theta|\mathbf{r}) = \frac{P(r_1, r_2, \ldots, r_m|\theta)p(\theta)}{P(r_1, r_2, \ldots, r_m)}.$$

In our case, however, we need a different update strategy that makes sense of the structured observations. We do not want to simply assume that we sample $\mathbf{r} = r_1, r_2, \ldots, r_m$ from the relevant population. We want also to take into account that $\mathbf{r}$ was the product of a certain user strategy. We also might have some knowledge of the cognitive process behind this strategy or of the presence of at least two agents, a human user who has been instructed to search (and that we know the rules in terms of a narrative or an experimental protocol) and a retrieval system which can be similar to the system that we use for our experiments. This rich structure of session records is not accounted for in a classic retrieval model.

We expect that the potential performance enhancement upon exploiting this structured information about a search process pays back for the added complexity that we must introduce in the relevance model. The major source of this additional complexity will appear to be the dependence between probability distributions and epistemic states: whereas, see for example [3, eq.5.5], in the standard generative model we need only one distribution over each representation, queries or documents, when we include actions, distributions will prolify, since we will generate one distribution for each agent in each possible epistemic state. This complexity, however, only reflects the complexity of the structure that comes with our observation. Standard relevance models are a distribution over the vocabulary, conditioned on the observation of a string: it is not surprising that if we want to introduce a probabilistic update upon observation of what we assume is a much more complicated process, the complexity of the formalism will inevitably increase.

At a minimum we need a probabilistic epistemic language $\mathcal{L}_{PEL}$ to express sentences about what agents believe, as on that depends their probability assignments. We also need a semantic to interpret these sentences and an update frame that, once combined with the prior model, outputs a new epistemic model with posterior probabilities. Hopefully the posterior distribution $p(\theta|\phi)$ so obtained is a better estimation of the unknown parameter θ_R than that from [3, eq.3.9].

In text retrieval, the set of primitive proposition that agents are supposed to reason about are of the type 'the relative frequency of term a in the relevant population is $f(a)$'. We assume that there is a search process that, after a certain number of steps which depend on a user's skills, will generate the relevant population $\mathcal{S}_R$; an optimally skilled user is someone who successfully pursues this path. In this case we do not need to represent sessions' interactions because the final query will generate the best possible ranking. However, while we do not have these idealised optimal users, the search skills of our users still improve during a search session.

We are interested in the probabilities that users assign at different stages of this learning process: this is the relevance feedback that our users indirectly provide to the system. We want also to provide a uniform account of actions as if they were documents and queries. In more formal terms, we are interested in the probability of the denotation of a formula in the language $\mathcal{L}_{PEL}$ that represents the result of the observed learning process. These are probabilities over the full event space $\mathcal{E}$, that is a σ-algebra over the full representation space $\mathcal{S}$: $\mathcal{E} \subseteq \wp(\mathcal{S})$

We propose an epistemic alternative to the pure probabilistic update

[3, eq.3.9] by first giving an epistemic interpretation of eq. 3.9, next to its frequentist and bayesian versions. Eq. 3.9 defines an update function $P(\cdot|\mathbf{r})$ for an agent who observes the statement that $\mathbf{r}$, which is a true statement so that $P(\mathbf{r}|\mathbf{r}) = 1$. We can represent a prior epistemic status as a network of states that are singled out by the truth values of different propositions at those states: accessibility relations connect states into an oriented graph representing what an agent believes to be true. In a typical IR application, for example, we might think of states as labeled by different configurations of a distribution parameter θ; whereas in standard epistemic logic there is a focus on factual knowledge, the world that is actually the case, in IR the focus will be on relevance.

Applying eq. 3.9 according to this interpretation amounts then to define a distribution μ over the set S of all possible epistemic states, such that a sum of μ over all states $s \in S$ is defined to be 1, and a discrete update function that maps a distribution before the observation that $\mathbf{r}$ to a distribution after that observation, as:

$$\mu'(s) = \begin{cases} \mu(s) \cdot \left(1 + \frac{P(\neg \mathbf{r})}{P(\mathbf{r})}\right), & \text{if } s \vDash \mathbf{r} \\ 0, & \text{if } s \nvDash \mathbf{r} \end{cases}$$

where $P(\mathbf{r}) = \sum_{\{s \in S | s \models \mathbf{r}\}} \mu(s)$ is the confidence in the observation that $\mathbf{r}$. The update from eq. 3.9 amounts then to eliminating all the epistemic states where the observed $\mathbf{r}$ does not hold and renormalising the distribution over the remaining states in the same proportions as before the observation.

This very same feature of updating with conditional probabilities was already questioned in other logical approaches to IR such as Logical Imaging [1] and its quantum IR version [9]. In the imaging approach renormalisation after an update was performed by considering that the probability mass taken away from $\neg\mathbf{r}$ states should be assigned only to states similar to the removed states, under some suitable similarity measure.

The alternative update based on PDEL considers the uncertainty in the learning process or *observation probabilities* [8] so that we cannot say that $\mathbf{r}$ is definitely a token of relevance. In our approach we also agree that renormalisation should not be uniform as in standard update, and in our contribution we aim at deriving from user models and observed interactions a set of *occurrence probabilites* for events, and use them to perform the update more effectively.

2.2 TREC evaluation

The retrieval model introduced in the previous section extends conditional probability with an account of how a user's interactions with a system changes the probability model. Information about interactions, in our case the session information, which belongs to each of the 4 TREC tasks, determine the space upon which the final system computes a standard conditional probability.

At the first step, before any interaction, the PDEL model contains only an initial distribution on the parameter space. This model accounts for the *prior probability*, one of the three kind of probabilities that we want to model.

To a second type of probability belong the *occurrence probabilities* for events. While in a real life situations [6] this probability reflects our knowledge of how information needs arise, in the TREC setting this probabilities model our knowledge of the experimental setting, that is represented in the procedure used to create the dataset. The main component of this experimental protocol is a narrative, a set of sentences designed to guide the user in finding relevant documents for the topic.

Occurrence probabilities for events are taken into account by means of a set Φ of preconditions and a function PRE that assigns to each element of Φ a probability distribution over the set of events $\mathcal{E}$. The PRE function determines the states of the updated model, allowing only states where a positive probability is assigned to the event at a state of the prior model. For example an updated model S' will contain a state (s', e) iff there is a state $s = \langle\theta_m\rangle$ labeled with a certain parameter θ_m in the prior model S and there is an event $e \in \mathcal{E}$ such that $\text{PRE}(s, e) > 0$. In the TREC application we consider only cooperative users: we only allow query events that are relevant to the global narrative, given the knowledge represented at each state.

For each distribution θ_s at state s we consider the difference under Kolmogorov-Smirnov test between θ_s and all the distributions $\theta', \ldots, \theta^m$ obtained by considering each sentence of the narrative to be a sample from the relevant population. The cardinality of the event set is equal to the cardinality of the set of narrative sentences. We then compare the effect of updating each distribution pair with the observed query: we define relevance as the property of reducing the gap between each distribution pair compared with a uninformative sentence represented by a sample from the background distribution. Therefore the PRE function will be positive if the difference exceeds a treshold that depends on the background probability.

We can interpret the preconditions in terms of the relevance model in [3, eq. 3.8] if we consider that each state is a point of the integration range Θ. The integral can be splitted in subintervals and multiplied by a constant before normalisation: some intervals of possible parameters can be eliminated if the constant is zero.

A third component of the updated model is the set of uncertainty relations between events, which models the *observation probabilities*. At the first step, before any query refinement, we take our best guess for these probabilities to be the effect, again compared to that of a sample of the background probabilities, of updating a pair of narrative distributions with the observed query. Notice that the model does not depend on a choice of the similarity test: we can use any estimation of the similarity between the ranked lists that an IR system would produce by exchanging the two probability distributions. The rationale is that if a system cannot produce different ranked lists when different prior models are updated with the a relevant sample, either the query is not discriminative enough or the collection does not contain enough information to effectively handle the query: in both cases this is a metric for the uncertainty about which distribution is the most appropriate to generate the relevant population.

The normalised product of these three probabilities yields the prior probabilities of the relevant model at the following step, that is the probabilistic uncertainties between the new states, now labeled with the content of the previous state (a distribution parameter) and the action taken (a query interpreted in one of the possible senses according to the narrative).

2.2.1 RL1 task: prior distribution

The prior model has n states, each labeled with one distribution parameter and probability $P_{t_syst,s} = \frac{1}{n}$. That is, from the test system perspective, each parameter is a priori an equally probable generator for the relevant population. We estimate the distributions at each state using the narrative: we use each sentence as a query and

we retrieve k_p documents. We tested this approach on 2011 data and we found out that retrieving more than 25 documents ensures that the estimated distributions are not too similar to each other. Each retrieved list determines a distribution over the entire session vocabulary, that is over all the terms that have nonzero probability at least in one state distribution. In order to avoid zero probabilities in a state distribution, we mix the estimated distribution with the background distribution under a mixing parameter $\lambda = 0.3$. We take the top e_p most frequent terms at each state as expansion for the current query at RL1. In TREC 2012 we limited the size of the distribution to $k_p = 3$ and of the query expansions to $e_p = 5$. A weighting schema assigns to the current query a weight equal to the cardinality of the state set, while state weights are inversely proportional to their normalised distance under K-S measure to the background distribution.

2.2.2 RL2-4 tasks: update with events

For the remaining tasks we can use the general form of the update rule in PDEL [8], that is:

$$P'_a((s,e),(s',e')) = \frac{P_a(s)(s') \cdot \text{PRE}(s',e') \cdot P_a(e)(e')}{\sum_{s''\in S, e''\in E} P_a(s)(s'') \cdot \text{PRE}(s'',e'') \cdot P_a(e)(e'')}$$

consisting of a re-normalised product of all the probabilities for events.

The queries q define in RL2 the event set $\mathcal{E}$. Each component of the event set defines a possible sense of the query. We hypothesise that each query sense can be mapped to a distribution associated with the relevance model after having observed the query, restricted to the original prior. As before the preconditions are the distances of each updated distribution to the distributions at the previous step. To estimate the observation probabilities, we first calculate the marginal distance of a state distribution to each other state distribution. This is a measure of the probability of confusing a query sense with another. We derive a total observation probability by summing onto the marginals.

In the RL3 task we consider the effect that the output of the retrieval system has on a user's epistemic state. We assume that browsing a ranked list amounts to creating an additional state that extends the original prior. Interactions can therefore, not only reweight some state, but also add new ones to the model. We recalculate the entire query update upon a new model, where the distributions, including those of the prior, now range onto a possibly larger session vocabulary.

In the RL4 task, clicks induce an update similar to queries. We use the top e_c most frequent terms from the snippets of the clicked documents to update the entire model as if they were query terms. This is a coarse approximation of using the term distributions from the whole clicked document, but given the limited session vocabulary, we expect that effect on the final rank will be negligible. We also limit the snippet size to $e_c = 5$.

2.3 Results and discussion

The left side of the table here below shows the nDCG at k of our proposed model. To the right side we report the median among the TREC participants.

Clearly this is only a preliminary assessment of our results based exclusively on the evaluation script output provided by the organisers.

Task	CWI submission	median TREC
RL1 (prior)	0.2422	0.1746
RL2 (query upd.)	0.2529	0.1901
RL3 (RL upd.)	0.2313	0.2160
RL4 (click upd.)	0.2319	0.2261

Table 1: nDCG at k of the CWI submission and median of TREC 2012 over all sessions

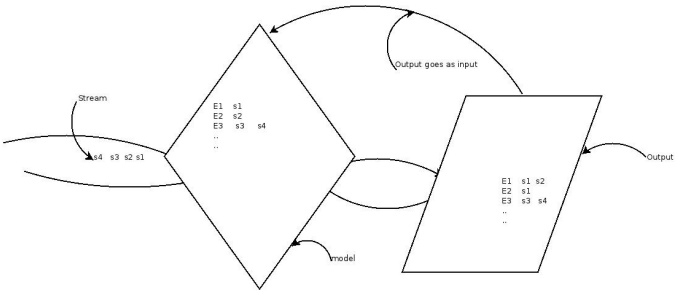


Figure 1: Initial approach

These results only allows to evaluate our choice of the prior and its associated weighting scheme for RL1. Our system has been explicilty designed to capture a user's learning process, that is how, starting from a possibly uninformative prior, the probability mass can be redistributed to states associated with the relevant topics. Aspects of the topic that a user learned to be less relevant should be discounted. Under the assumption of a moderately effective user, who learns as a session goes on, high probable states in the model should correspond to the relevant distribution of the current query.

We interpret the high score on all tasks, combined with a negligible difference between tasks, to the nature of the topics. They require just few interactions to be resolved and test users show a good understanding of what kind of documents might possibly be used to accomplish these tasks. The query update rule improves on the prior, indicating that queries are the most effective mode of interaction, hence a great deal of the uncertainty is in how to issue the correct query terms in this particular system.

3. KNOWLEDGE BASE ACCELERATION

3.1 Intial ideas on approaches and architecture

It was stated that 4% of the Wikipedia citations did not mention the Wikipedia entities they are cited by. We thought there could be more documents in the stream that do not mention the Wikipedia entities by name and yet are relevant. Capturing relevant documents that do not mention entities by name was our first interest. At this moment, we wanted to capture two things:

- Stream documents that do not mention Wikipedia entities by name and yet are relevant(central), and
- The evolution of an entity as documents that are relevant(central) are found

To accomodate these two interests, we thought of an approach as in Figure 3.1.

The model could be anything. We thought of a machine-learned model, string similarity, string-matching, or any other. The output was taken as input because we thought the evolution of an entity has

changed the entity and so the new representation should be used in the query. The big questions here were

- How to represent the WP entities and the stream-documents
- What to add to the representation of the Wikipedia entities when a relevant stream document is found
- What models, approaches to use

3.1.1 Release of sample annotation and some statistics

	garbage	neutral	relevant	central	total
mention	7991	3862	13971	7806	33630
not mentions	15367	163	61	0	15591

Table 2: statistics from few weeks annotation

Statistics from the few weeks annotation changed the way we see the task. Out of all non-mentioning documents, only 0.4% are relevant, 0% are central. So we did not see, from a performance perspective, a point in concentrating our efforts on detecting non-mentioning-yet-relevant stream documents. Instead, we decided to focus only on mentioning and relevant or/and central. The implication of this is that the two big questions we raised above are no longer important. The reason why the first question is not important is evident. The second question is also the same because for a document to be relevant or central, it will almost always mention the entity, thus no need to update representation.

3.1.2 A new challenge and a new approach

Out of all mentioning documents, 23.8% are garbage, 35.3% are garbage or neutral. Now, the challenge is not how to filter non-mentioning-yet relevant, but how to exclude mentioning-yet-non-relevant i.e. garbage and neutral. Another challenge is that the entities are ambiguous in the sense that two entities can have the same or nearly the same name, and thus the same representation. This observation informed our next choices of approaches. We thought of approaches that can, at least, solve one of the two problems. Our supervised approaches attempt to solve both problems while our unsupervised ones mainly attempt to solve the ambiguity problem. When we were pondering about solutions to this, we came across a resource called Google-Cross-Lingual dictionary (GCLD). We thought that using the GCLD captures the second problem since it has probabilities for the strings and concepts. We also thought that, in combination with other disambiguation methods, the GCLD could be used to distinguish garbage and neutral from relevant and central.

Therefore the approach now is exactly like Figure 3.1, but without the output going as input. This means to first represent the queries (the Wikipedia entities) in some way and to query the stream. After finding a match of the strings for an entity in the stream, we use the probabilities to give a confidence score. All our approaches revolve around the choices of entity representation, the scoring function to measure confidence and scaling functions.

3.2 Representation

Representing the Wikipedia entities (the queries) and the stream documents in some way is mandatory. At first, we thought we can represent the streams in terms of n-gram tokens thereby reducing the size of the corpus, but then that would confine us to only some approaches that can consume the tokens. So we left stream documents representation to simple representation during processing. However, we needed to represent the Wikipedia entities in some way. All our approaches used a different entity representation and that is the main component. The components of any of our approaches are entity representation, string matching, scoring and, to some extent, scaling functions.

3.3 Development Environment

We used JAVA and python as main programming languages. We used Hadoop architecture provided by SARA (the Natherlands SARA Computing and Networking Services) and Java to process the data in a map reduce architecture. Python was used to process the same data in a federated fashion: six computers each with a 8-core processors and GNU tool called GNU Parallel to parallalize the process. The Java Hadoop architecture was much faster than the federated architecture.

3.4 Supervised Approaches

We describe now the two supervised approaches used in this challenge. The first approach, Prefix-Suffix Learning, focuses on precision, penalizing the recall. The second, Language Model, focuses on balancing the precision and recall.

3.4.1 Prefix-Suffix Learning Approach

In this approach, for each entity e, we learned a set of strings S_e of the form uv and vw that occur in the documents annotated as central (denoted as Δ^+) and does not occur in document annotated as relevant, neutral or garbage (denoted as Δ^-); where $v \in V_e$, u is prefix of v of size K and w is a suffix of v of size K. We vary K in the interval $[1, .., K]$, then we learned at most $2K$ different strings per unique $v \in V_e$.

Example: Consider the entity Nassim Taleb, its set $V_{NassimTaleb} = \{$*Nassim Nicholas Taleb, Nassim Taleb*$\}$ and the news document below:

"Among the people we reached out to while reporting this week's cover story on Rich Marin ***was Nassim Taleb. Not*** *only is he a well-known talking head (and presumably accessible), but he also got his start at Bankers Trust, just like Mr. Marin. "*

For $K = 4$, the set $S_{NassimTaleb} = \{$ *Nassim Taleb, s Nassim Taleb, as Nassim Taleb, was Nassim Taleb, Nassim Taleb., Nassim Taleb. , Nassim Taleb. N, Nassim Taleb. No*$\}$

Then, for each entity e, an arbitrary document D is annotated as central if any string in S_e occur in D.

The algorithm to learn the string S_e is described in the Alg.1. In the run that we submitted, we used $K = 10$.

3.4.2 Language Model Approach

This approach focuses on balancing precision and recall. To do so, we build a *statistic language model* for each entity $e \in E$ using the training documents annotated as central. Then we score the documents based on the *perplexity measure* between the entity language model and the document text. We normalize the values between [0,1000]. This measure may produce some documents with low score; however, we consider all those documents with score > 0 as central. Below we detail this approach.

DEFINITION 1 (ENTITY LANGUAGE MODEL). *Given an entity e and a corpora Δ^+ of documents annotated as central, we build a statistical language model, trigram model, LM_e for e over $\bigcup_{d \in \Delta_e^+} d$, where $\Delta_e^+ = \{d | \forall d \in \Delta^+ \wedge v \in V_e \wedge d = uvw\}$*

Algorithm 1 LearningPrefixSuffixStrings(E, Δ^+,Δ^-, K).

```
S ← ∅
for all e ∈ E do
  S_e ← ∅
  V_e ← QueryDBPediaLabels(e)
  for all v ∈ V_e do
    for all d ∈ Δ+ do
      if d.contains(v) then
        S_e ← S_e ∪ prefix(v, d, K)
        S_e ← S_e ∪ suffix(v, d, K)
      end if
    end for
    for all d ∈ Δ− do
      if d.contains(v) then
        N_e ← N_e ∪ prefix(v, d, K)
        N_e ← N_e ∪ suffix(v, d, K)
      end if
    end for
  end for
  S ← S ∪ (S_e − N_e)
end for
return S
```

In other words, for each entity e we build a trigram language model LM_e over an aggregate document containing all document annotated as central for the entity e.

We particularly used the *Kyoto Language Modeling Toolkit (Kylm)*[1] to build this language model. The only parameter set in this api was `smoothuni`. Alg.3 describe the process of creating the language model.

Algorithm 2 LanguageModelTraining(E, Δ^+).

```
LM ← ∅
for all e ∈ E do
  V_e ← QueryDBPediaLabels(e)
  for all v ∈ V_e do
    for all d ∈ Δ+ do
      if d.contains(v) then
        Δ+_e ← Δ+_e ∪ d
      end if
    end for
  end for
  LM_e ← NGramModel(Δ+_e, 3) #Trigram
  LM ← LM ∪ LM_e
end for
return LM
```

In order to annotated the test corpora as central, we compute the perplexity between a new document and each specific language model.

The perplexity is based on entropy, where the entropy gives a measure of how likely the ngram model is to have generated the test data. Entropy is defined (for a sliding-window type ngram) as:

$$H = -\frac{1}{Q}\sum_{i=1}^{Q} logP(w_i|w_{i-1}, w_{i-2}, ...w_{i-N+1})$$

where Q is the number of words of test data and N is the order of the ngram model. Perplexity is a more intuitive measure, defined as:

$$perplexity = 2^H$$

The perplexity of an ngram model with vocabulary size W will be between 1 and W. Low perplexity indicates a more predictable language. All documents with perplexity < 100 were consider as central.

9[1]http://www.phontron.com/kylm/

Alg.4 describes this process.

Algorithm 3 LanguageModelCentralAnnotator(E, D, LM).

```
CENTRAL ← newMatrix(D, E)
for all e ∈ E do
  V_e ← QueryDBPediaLabels(e)
  for all v ∈ V_e do
    for all d ∈ D do
      if d.contains(v) then
        p ← perplexity(LM_e, d)
        if p < 100 then
          CENTRAL[d][e] ← p
        end if
      end if
    end for
  end for
end for
return CENTRAL
```

3.5 Unsupervised Approaches

We have experimented with two main unsupervised approaches: one so-called disambiguator and another so-called Google-Cross-Lingual Dictionary (GCLD). In both of them, The Wikipedia Wikipedia entities are represented by strings and those strings are used to query the stream. If a matching string is found, then the document is relevant and/or central to a degree provided by a scoring function. The disambiguator targets the central bins while the GCLD targets the relevant and central bins. Under the GCLD based approach, we have experimented with many variations of scoring functions, thresholds and scaling functions.

3.5.1 Disambiguator

Entity Representation

Given an entity e among the 29 given Wikipedia entities, we represent e as a set of strings V_e, defined as:

DEFINITION 2 (ENTITY REPRESENTATION). *An entity representation V_e of an entity e is the result of the SPARQL*[2] *query over DBPedia*[3] *representation of an entity e - a RDF version of Wikipedia:*

```
SELECT distinct ?o WHERE
{ e <http://www.w3.org/2000/01/rdf-schema#label> ?o .}

UNION

SELECT distinct ?o WHERE {
e <http://xmlns.com/foaf/0.1/name> ?o .}
```

In other words, the set V_e is the set of `labels` and `name` of the entity e in DBPedia. For example, the set V for the entity Nassim Taleb [4], is: $V_{NassimTaleb} = \{$*Nassim Nicholas Taleb, Nassim Taleb*$\}$

Disambiguator Approach

This approach aims at producing high recall and improve the precision over the baseline by solving only the cases where the source entities are ambiguous.

DEFINITION 3 (AMBIGUOUS ENTITIES). *Given two distinct entities e and f, they are ambiguous if their entity representations $V_e \cap V_f \neq \emptyset$.*

9[2]http://en.wikipedia.org/wiki/SPARQL

9[3]http://dbpedia.org

9[4]In DBPedia: http://dbpedia.org/page/Nassim_Nicholas_Taleb

Basically, to produce high recall, we select the documents using the string in the entity representation V_e, and then to improve precision over this initial selection, we filter the documents using an extended entity representation T_e that we describe next.

When two or more entities are ambiguous, it is impossible to decide what exactly entity a string mentioned in a document refers to. For example, among the 29 entities provided in TREC-KBA 2012, four entities were ambiguous:

```
Boris_Berezovsky_(businessman);
Boris_Berezovsky_(pianist);
Basic_Element_(company);
Basic_Element_(music_group).
```

For example, the string *Boris Berezovsky* may refer to a pianist or a businessman. Without context information, we cannot decide which one it refers to.

In order to decide which one of the ambiguous entity an document mention, we contextualize an entity representation using type information extracted from DBPedia representation of this entity. An t*ypified entity representation* is defined such as:

DEFINITION 4 (TYPIFIED ENTITY REPRESENTATION). *A typified entity representation T_e of an entity e is the results of the SPARQL query below:*

```
SELECT distinct ?c WHERE {
e <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }

UNION

SELECT distinct ?c WHERE {
e <http://purl.org/dc/terms/subject> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }

UNION

SELECT distinct ?c WHERE {
e <http://purl.org/dc/terms/subject> ?z .
?z <http://www.w3.org/2004/02/skos/core#broader> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }
```

In other words, this query retrieves a set of string representing the type of an entity e in its DBPedia representation. The property `type` and `subject` above define the type of a specific DBPedia representation. For example, considering the entity *Boris_Berezovsky_(businessman)*, the instantiation of the query above would be:

```
SELECT distinct ?c WHERE {
<http://dbpedia.org/page/Boris_Berezovsky_(pianist)>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }

UNION

SELECT distinct ?c WHERE {
<http://dbpedia.org/page/Boris_Berezovsky_(pianist)>
<http://purl.org/dc/terms/subject> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }

UNION

SELECT distinct ?c WHERE {
<http://dbpedia.org/page/Boris_Berezovsky_(pianist)>
<http://purl.org/dc/terms/subject> ?z .
?z <http://www.w3.org/2004/02/skos/core#broader> ?o .
?o <http://www.w3.org/2000/01/rdf-schema#label> ?c. }
```

Then $T_{Boris\_Berezovsky\_(pianist)} = \{pianist, musician, russian\}$.

Given an entity e, the algorithm process the news stream sequentially and for all document D that contains a string $v \in V_e$, we produce a *Jaccard score* between T_e and d_e, where d_e and *Jaccard* is defined such as:

DEFINITION 5 (ENTITY CONTEXT). *A entity context d_e of an entity e is a set of tokens of a string uvw in D, where $u \in V_e$, and u and w have at most a length L. In the run that we submitted, we set $L = 400$.*

DEFINITION 6 (JACCARD SCORE). *Given two sets A and B, the Jaccard score of these sets are:*

$$Jaccard = \frac{(A \cap B)}{(A \cup B)}$$

A document D is considered central for an entity e if:

$Jaccard(T_e, d_e) > 0 \wedge \forall x \in E : x \neq e, Jaccard(T_e, d_e) > Jaccard(T_x, d_x)$

Alg. 2 describes the process of annotating central documents using this method.

Algorithm 4 DisambiguatorCentralAnnotator(E, D, L).

```
CENTRAL ← ∅
MATRIX ← newMatrix(D, E)
for all e ∈ E do
  V_e ← QueryDBPediaLabels(e)
  T_e ← QueryDBPediaTypes(e)
  for all v ∈ V_e do
    for all d ∈ D do
      if d.contains(v) then
        d_e ← context(d, e, L)
        MATRIX[d][e] ← MATRIX[d][e] + Jaccard(T_e, d_e)
      end if
    end for
  end for
end for
for all d ∈ D do
  if MATRIX[d].max > 0 then
    CENTRAL ← CENTRAL ∪ [d, MATRIX[d].maxEntity]
  end if
end for
return CENTRAL
```

3.5.2 Google Cross Lingual Dictionary (GCLD) Approach

In our main unsupervised approach, we use a resource called Google-Cross-Lingual dictionary (GCLD) that maps language independent strings of words and Wikipedia articles (also called concepts or URLs). The resource assigns empirical probability distributions to strings given a URL and to URLs given a string [7]. Our approach here is to represent the queries (the Wikipedia entities) by the strings in the dictionary and to use the new representation as a query to filter documents that are central or/and relevant from the stream. The probabilities are used to give a confidence score for the relevance of a document for a Wikipedia entity. The dictionary has many different statistical information that can be used in different ways to improve performance and we have tried to experiment with some and examined their effects. Below we will detail the dictionary, the approach we used, the experiments we did and discuss the results and draw conclusions.

The dictionary

tf-IDF	ditionary
t =term	l(s,e) - an instance of a link between anchor s and WP e
tf = term-frequency d	#l(s,e) - the total number of hyperlinks to a Wikipedia article having s as anchor $\sum_{s \in S} l(s,e)$-all links to a WP article
df_t	The total number of links that contain s, #$\sum_{s \in S} l(s,e)$ that contain particular l(s,e), $lf_{l(s,e)}$
N	#$\sum_{s \in S} l(s,e)$ - the collection, the number of WP entities in this case
$idf_t = log \frac{N}{df_t}$	$idf_{l(s,e)} = \frac{\#l(s,e)}{\sum_{s \in S} l(s,e)}$

Table 3: Analogy to tf-IDF table

The dictionary is bi-directional in the sense it provides a mapping from free-form-natural language strings to concepts and vice versa. The strings are gathered from anchor texts to all Wikipedia pages and the English Wikipedia titles. This means the strings include anchor texts from inter-wikipedia linking, and anchor texts to non-English Wikipedia articles. The strength of association between strings and concepts is quantified by conditional probabilities.

Let $s \in S$ be a string and let $e \in E$ be a Wikipedia entity. $l(s,e)$ is a link between s and e where s is used as an anchor in a link to a Wikipedia entity e. $\#l(s,e)$ is the total number of hyperlinks into a Wikipedia article e using anchor text s. $\sum_{e \in E} l(s,e)$ is the total number of links into Wikipedia pages that use s as an anchor and $\sum_{s \in S} l(s,e)$ is the total number of links to a Wikipedia article e. Based on this, the dictionary defines two probabilities : $P(URL|s)$ for strings to concepts and $P(s|URL)$ for concepts to strings as follows.

$$P(URL|s) = \frac{\#l(s,e)}{\sum_{e \in E} l(s,e)} \quad (1)$$

$$P(s|URL) = \frac{\#l(s,e)}{\sum_{s \in S} l(s,e)} \quad (2)$$

Formula one tells whether a string is ever used as an anchor text to a certain Wikipedia entity and if it does it gives the probability. By this, it disambiguates the string by distributing the probability mass over the different Wikipedia entities according to how often it is used as an anchor to each of them. The second formula quantifies how important as an anchor a certain string is in comparison to other strings that also point to the same entity e. All strings that can be used as anchors to a certain Wikipedia article are co-referents, and the second formula measures the relative strength with which a co-referent refers to a Wikipedia article.

An alternative way to look at the two formulas is to interpret them analogous to the tf-IDF concept (see table 3). Analogous to the tf-IDF concept, a document is the number of links pointing to a WP article.

One can think that the first formula is like the term frequency normalized by the number of terms and the second formula is a modified idf, i.e. it uses the number of links into a document instead of the number of the collection and normalizes it by the number of all links having anchor s.

Experiments, Results and discussions

We conducted many experiments by varying dictionary strings for representation, probabilities for scoring, and thresholds for selecting strings. The algorithm is string matching, i.e. once the Wikipedia entities are represented with our choice of set of strings, we query each document of the stream if it has a match for the elements of our set. If there is a match, we give the document-entity pair a confidence score computed based on the probabilities. When more than one element of the set of strings for an entity are matched, we take either the average or the maximum of the probabilities of the matched strings. Our measures were recall, precision, and F-measure against relevance cut-off. But to distinguish between two approaches, we mainly looked at F-measure.

Our first experiment was with probabilities given by $P(URL|s)$. We lowercased all the string representations of the Wikipedia entities and the stream documents. When two strings are lowercased to the same form, we assign the form we keep the higher probability. We also stripped punctuation and white spaces. We did experiments with strings that come only from non-Wikipedia pages, and all strings to English or corresponding non-English Wikipedia pages. We compared the results on F-measure and the later representation performed better. The reason for increment in F-measure was because of an increase in recall. And the increase in recall is due to the additional strings. Using average of the probabilities of the matching strings performed worse than the maximum. Next, we experimented by setting different thresholds on probabilities in order to select strings that have higher probabilities. We tried thresholds 0.01, 0.001, 0.0001. However, the performance did not improve significantly. In fact, in the case of threshold 0.01, performance dropped significantly.

Our second experiment was in lowercasing and stripping punctuations. The dictionary strings that we used are not lowercased, i.e. “Nasim” and “NASIM” are considered different strings. The dictionary strings also contain punctuations and white spaces. We decided to experiment without lowercasing the entity representations and the stream documents. The performance was a big improvement over the lowercased and punctuation-stripped approach. It is not surprising that it is so since it better captures the capitalizations which are a feature of proper nouns.

However, we were not happy with results since the performance scores were still poor. Moreover, thus the confidence scores were very small and were very susceptible to scaling functions. $P(URL|s)$ is like a tf, it never tells us how discriminative a string is to a certain Wikipedia entity with respect to other Wikipedia entities. $P(s|URL)$ is the right probability to use for this purpose. $P(s|URL)$ exposes the ambiguity in a string by distributing the probability mass over the entities it can be used as anchor in a link. There are many strings whose $P(s|URL)$ probabilities were 1, which shows that the document containing the string is highly probably relevant to the WP entity the string represents. And, indeed, experiments using these probabilities for scoring showed better performance. The use of $P(s|URL)$ disambiguates ambiguous entities naturally. Varying thresholds for string selection and using averaging instead of maximum did not improve results significantly.

Our third experiment was combining the two probabilities for scoring. When more than one string is matched, we multiply both probabilities first, and keep the maximum as a score. Our best scores on the relevant and central bins was obtained by this approach. Our main run submissions were from this approach. We also submitted

runs using the same approach but with lowecased and punctuation-stripped. For both cases, we used two different simple per-entity scaling functions. First, we selected the maximum score per entity and use that to scale the results as:

$$s_scaled(doc - entity) = \frac{s(doc - entity)}{s_max(entity)} * 100\%. \quad (3)$$

Our second scaling function was using a threshold on maximum score per entity to discourage entities whose maximum score is less than 10. we used a threshold of 10 for $s_max(entity$ such that those entities whose highest score is less than 10 will be divided by 10.

Using the combined scoring with and without lowercasing and stripping and the two scaling functions, we submitted 4 runs. The non-stripping results were good. In total, we submitted 7 runs (1 prefix-suffix, 1 language model, 1 disambiguator, 4 GCLDbased). The GCLD submissions are google_dic_1(no lowercasing and no stripping off, normalizing by the highest score except those whose score is less than 10), google_dic_3(lowercasing and stripping off normalized by the highest score) , google_strip_1 (lowercasing stripping off, normalize by highest score except those whose score is less than 10) and google_strip_2(lowercasing and stripping off, normalizing by highest score).

GCLD challenges

The GCLD is a mapping between strings and Wikipedia concepts or vice versa. While the probabilities show how likely a string can be used as an anchor in a link to a Wikipedia page, it never shows how important the anchor text is for a document. The only relationship between the strings in the dictionary and the strings in the document is through string matching. This means a word may have a high probability of being used in a link to a Wikipedia page, but if the word is not important for the document, say Obama in a restaurant's name, the match becomes useless. We believe incorporating some third function that measures the importance of a term for a document can improve the performance. This is something we want to try next. Another challenge is the presence of noise in the GCLD. Strings such as "here" are present in many of the entity representations.

3.5.3 Run Graphs and comparisons

Figures 2 and 3 show the performance on the test set of disambiguator and prefix-suffix learning respectively. Similarly, Tables 4 and 5 show the two best performing variations of GCLD. 6 shows the highest score for each entity and the run that generated it.

4. CONCLUSIONS

In the session track, we experimented with our system that is explicitly designed to capture a user's learning process. It achieves high scores on all tasks with a negligible difference between tasks. In KBA track, we have experimented with supervised and unsupervised approaches. Under the supervised approach, we have experimented with prefix-suffix learning and language models. On the training set, both learning approaches have comparable performance. Under the unsupervised approach, we have experimented with DBpedia labels and GCLD. Under GCLD, we have tried different entity representations, different scoring functions, and different scaling functions. The two supervised approaches and the DBpedia labels approach target the central bin. In the GCLD, we have targeted the central+relevant bins because the nature of the strings is not in a position to differentiate between central and relevant. On the test set, the DbPedia labels approach showed a better performance. On the central+relevant bins, the GCLD with no

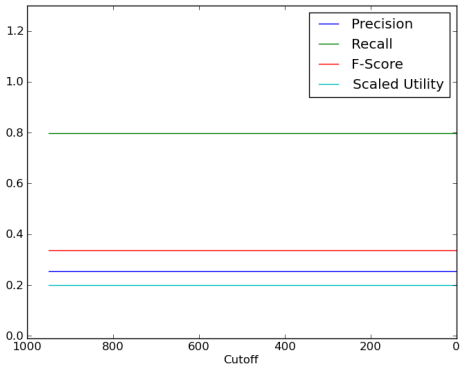


Figure 2: DbPedia.

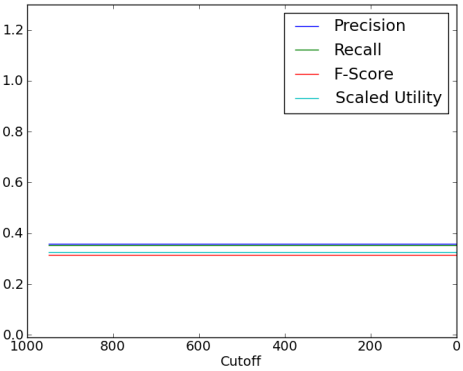


Figure 3: Prefix-sufix learning.

stripping and full per-entity normalization performed better than other variations.

Our experiments and approaches show that there are three factors that affect commulative citation recommendation: entity reperesentation, local context, and scoring. The GCLD and DbPedia representations and their better performance over other machine learning approaches testify that a good entity representation is important for CCR. A very important point about entity representation learned from the GCLD is that the entity representation should be used as they are i.e. without lowercasing and stripping off punctuations. The GCLD is noisy, but it also includes many of the DbPedia labels. Trec 2012 participant University of Delaware group has obtained a good result using entity name and related entity name.It is interesting to explore which entity representation is best and why for the future. The string learning performed well at detecting documents relevant to some ambigious entities such as the $Basicelements$ suggesting that local context can help improve results. The importance of a scoring function is shown by the different scoring functions that we used in GCLD. For the future, studying the interaction between entity representation, local context and scoring functions may give a better insight into what constitues a better CCR system.

5. REFERENCES

[1] F. Crestani and C. J. V. Rijsbergen. Information retrieval by logical imaging. *Journal of Documentation*, 51:3–17, 1995.

[2] E. T. Jaynes. *Probability Theory : The Logic of Science*. Cambridge University Press, April 2003.

[3] V. Lavrenko. *A generative theory of relevance*. PhD thesis,

Entities	Central		Relevant+central	
	maxF	approach	maxF	approach
Mario_Garnero	0.952	google_dic_1, google_dic_3	0.952	google_dic_3, google_dic_1
Basic_Element_(company)	0.870	prefix_suffix	0.594	google_dic_3, google_dic_1
Basic_Element_(music_group)	0.731	prefix_suffix	0.038	google_dic_3, google_dic_1
Satoshi_Ishii	0.642	disambiguation	0.604	google_dic_3, google_dic_1
Jim_Steyer	0.634	disambiguation	0.0	all
Ikuhisa_Minowa	0.612	google_dic_1, google_dic_3	0.612	google_dic_3 , google_dic_1
William_D._Cohan	0.611	disambiguation	0.395	google_dic_3, google_dic_1
Vladimir_Potanin	0.565	language_model	0.336	google_dic_3, google_dic_1
Boris_Berezovsky_(businessman)	0.544	google_dic_1, google_dic_3	0.544	google_dic_3, google_dic_1
Roustam_Tariko	0.471	disambiguation	0.466	google_dic_3, google_dic_1
Nassim_Nicholas_Taleb	0.466	disambiguation	0.463	google_dic_3, google_dic_1
Lisa_Bloom	0.451	prefix_suffix	0.153	google_dic_3, google_dic_1
Charlie_Savage	0.4	language_model	0.158	google_dic_3, google_dic_1
Annie_Laurie_Gaylor	0.396	disambiguation	0.392	google_dic_3, google_dic_1
Ruth_Rendell	0.392	disambiguation	0.386	google_dic_3, google_dic_1
Frederick_M._Lawrence	0.333	google_dic_3, disambiguation	0.333	google_dic_3, google_dic_1
James_McCartney	0.310	disambiguation	0.281	google_dic_3, google_dic_1
Alex_Kapranos	0.301	disambiguation	0.298	google_dic_3, google_dic_1
Darren_Rowse	0.290	disambiguation	0.270	google_dic_3, google_dic_1
Douglas_Carswell	0.235	prefix_suffix	0.162	google_dic_3, google_dic_1
Bill_Coen	0.235	disambiguation	0.231	google_dic_3, google_dic_1
Alexander_McCall_Smith	0.228	prefix_suffix	0.221	google_dic_3, google_dic_1
Aharon_Barak	0.184	disambiguation	0.183	google_dic_3, google_dic_1
Lovebug_Starski	0.167	language_model	0.125	google_dic_3, google_dic_1
William_H._Gates_sr	0.163	google_dic_1,google_dic_3	0.163	google_dic_3, google_dic_1
William_Cohen	0.147	google_strip_2, google_strip_2	0.147	google_strip_2, google_strip_1
Boris_Berezovsky_(pianist)	0.143	google_dic_1, google_dic_3	0.143	google_dic_3, google_dic_1
Rodrigo_Pimentel	0.114	disambiguation	0.047	google_dic_3, google_dic_1
Masaru_Emoto	0.104	google_strip_2,google_strip_2	0.104	google_strip_2, google_strip_1

Table 6: The highest maxF scores for each entity and the run that produced it (Results sorted by the maxF of the central column)

University of Massachusetts Amherst, 2004. AAI3152722.

[4] R. Nallapati, T. Minka, H. Zaragoza, and S. Robertson. The smoothed dirichlet distribution: Explaining kl-divergence in information retrieval. Technical report, CIIR, 2006.

[5] I. Ruthven and M. Lalmas. Using dempster-shafer's theory of evidence to combine aspects of information use. *Journal of Intelligent Information Systems*, 19:267–301, 2002. 10.1023/A:1020114205638.

[6] R. Savolainen. *Everyday information practices: a social phenomenological perspective*. Scarecrow Press, Lanham, Md., 2009.

[7] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for English Wikipedia concepts. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May 2012.

[8] J. van Benthem. Conditional probability meets update logic. *Journal of Logic, Language and Information*, 12:409–421, 2003. 10.1023/A:1025002917675.

[9] G. Zuccon, L. Azzopardi, and C. J. van Rijsbergen. Revisiting logical imaging for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 766–767, New York, NY, USA, 2009. ACM.

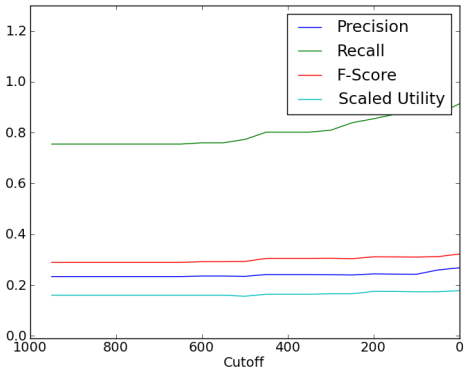
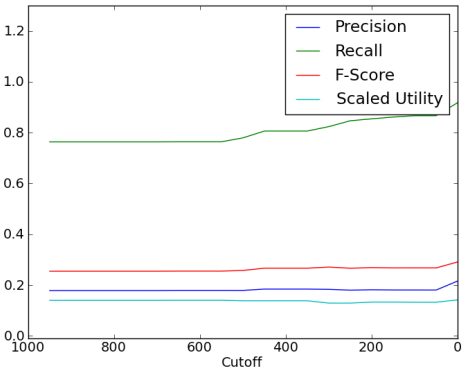
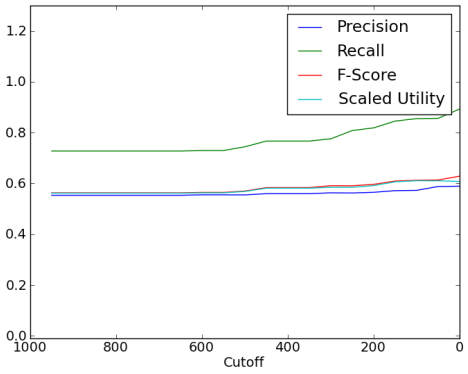
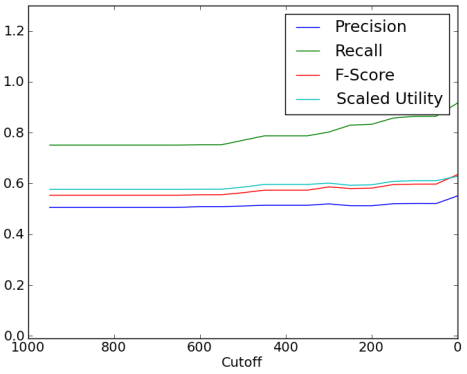
	cross_ligual_no_stripping_normalized_by_highest_score_per_entity	
	Training	Test
c		
c+r		

Table 4: Results of GCLD no stripping normalized by highest score. Training results in the left column and testing results in the right column.

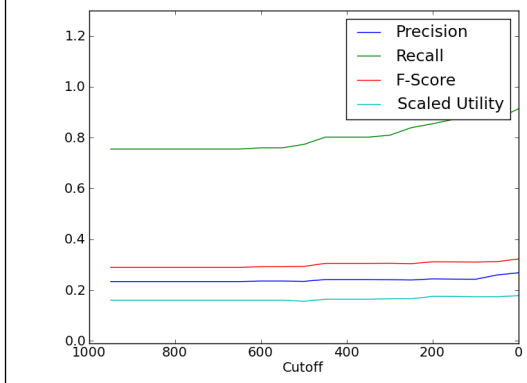
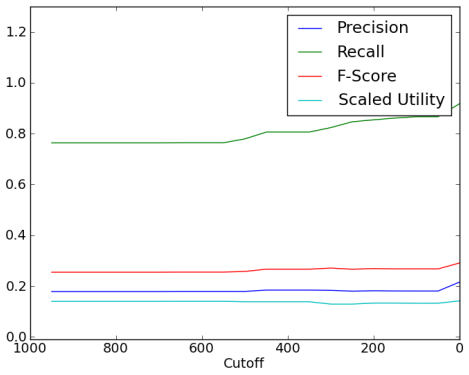
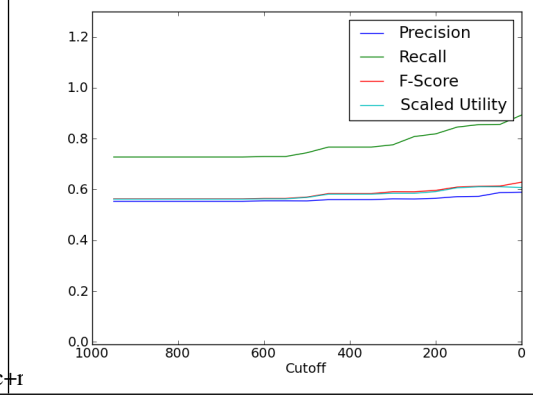
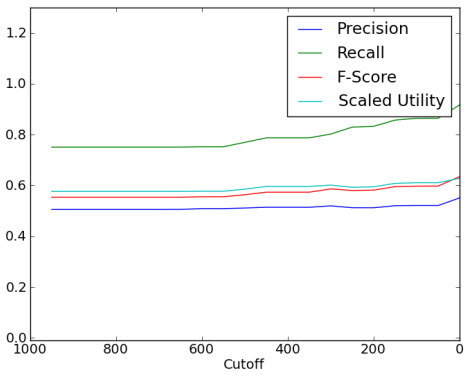
cross_ligual_no_stripping_normalized_by_highest_score_per_entity_except_those_less_then_10		
	Trainig	Test
c		
c+r		

Table 5: Results of GCLD no stripping normalized by highest score except those less than 10. Training results in the left column and testing results in the right column.