Overview of the TREC-2014 Microblog Track  
(Notebook Draft)

Jimmy Lin, 🟠 Miles Efron, 🟡 Yulu Wang, 🟠 and Garrick Sherman 🟡

üler of Maryland, College Park
Universiti of Illinois, Urbana-Champaign

jimmylin@umd.edu, mefron@illinois.edu, ylwang@cs.umd.edu, gsherma2@illinois.edu

1. INTRODUCTION

This year represents the fourth iteration of the TREC Microblog track, which has been running since 2011. The track continued using the “evaluation as a service” model [8, 7], in which participants had access to the document collection only through an API. In addition to the temporally-anchored ad hoc retrieval task, which has been running since the inception of the track, we introduced a new task called tweet timeline generation (TTG), where the goal is to produce concise “summaries” about a particular topic for human consumption.

Although this overview covers both tasks, more emphasis is placed on the tweet timeline generation task, which necessitated the development of a new evaluation methodology. For convenience, we refer to previous track overview papers [8, 12, 9] for details on the setup of the ad hoc task.

2. TASK DESCRIPTION

One assumption of the Cranfield Paradigm [4], which provides the basis of many TREC tracks, is that researchers can acquire the document collection under study. What if this were not possible? Tweets represent an example of such collections: Twitter’s terms of service forbid redistribution of tweets, and thus it would not be permissible for an organization to host a collection of tweets for download.

The evaluation-as-a-service (EaaS) model was introduced in the TREC 2013 Microblog track as one solution to this challenge [8, 7]. Under this model, we (as the track organizers) gathered a collection of tweets centrally, but, instead of distributing the tweets themselves, we provided a service API (built on the open-source Lucene search engine) through which participants could access the tweets to complete the task.

The collection and service API used in this year’s track was the same as last year’s [8]. The collection, known as Tweets2013, consists of 243 million tweets crawled from the public Twitter sample stream between February 1 and March 31, 2013 (inclusive). This level of access is available to anyone with a Twitter account and does not require any special authorization. The API provides basic search access and returns tweet content as well as various metadata fields. More details about the API specification are provided in last year’s track overview paper [8].

Experience from last year’s track suggests that the evaluation-as-a-service model provided a satisfactory solution to the collection redistribution issues. The level of participation (one of the most popular tracks) seems to indicate that the transition to API-based access was not overly burden-some, and that the API provided sufficient flexibility for participants to pursue their own research agendas.

2.1 Temporally-Anchored Ad Hoc Retrieval

The putative user model for the ad hoc retrieval task is as follows: “At time $T$, give me the most relevant tweets about an information need expressed as query $Q$.” Although the task definition has not substantively changed since the first Microblog track in TREC 2011, we have in recent years gained a more refined understanding of how the task is operationalized in TREC [16].

The above user model can actually describe two slightly different scenarios: Consider a scenario where a journalist is investigating a sports scandal that has been brewing for the past several weeks. She just got news of a breaking development, and turns to searching tweets to find out more details: the scandal’s major facts, reactions from fellow athletes, commentary from analysts, etc. Since this particular news story has been developing for several weeks, any keyword search involving the athlete’s name would likely bring up results from many different points in time. In this case, the journalist specifically wants to see the most recent tweets about the event. For convenience, we refer to this as the “real-time search” scenario.

Consider another scenario where a journalist is searching an archive of tweets as part of a retrospective piece on the impact of social media on the course of the Egyptian revolution. The topic is temporally-anchored in the sense that political events may not have played out fully, and the journalist is interested in perspectives at a particular point in time $T$. In this case, she might want to search for results that accurately reflect the volume of discourse on the topic (not necessarily biased toward more recent tweets prior to time $T$). For convenience, we refer to this as “archive search”.

Although the original conception of the task was closer to the real-time search scenario, NIST assessors indicate that topic development and relevance judgments followed a model closer to the archive search scenario. Given that the document collection was indeed retrospective, archive search represented a more natural operationalization of the ad hoc task.

From a technical perspective, these two alternative scenarios hold implications for the design of the search infrastructure. The search API is built on an index of the entire collection; the temporal constraint $T$ is satisfied by retrieving from the entire collection, but then discarding tweets that occur after $T$. One concern with this approach is that the service is taking advantage of term statistics of tweets
that occur “in the future” (if one wishes to simulate the real-time scenario). However, we have confirmed in a recent study that such inclusion of future term statistics does not substantively affect ranking results [16].

2.2 Tweet Timeline Generation

This year, we introduced a new task called tweet timeline generation (TTG), motivated by the observation that for many queries, search systems often return many tweets that are duplicates, near-duplicates, or contain the same (or highly-similar) information—a user is unlikely to want to see an enumeration of all these tweets. Instead, it would be desirable if the system produced a “summary” timeline about the topic. In the task definition this year, a summary is operationalized as a list of non-redundant, chronologically ordered tweets. Thus, the putative user model is as follows: “At time T, I have an information need expressed by query Q, and I would like a summary that captures relevant information.” It is imagined that the user would consume the entire summary (unlike a ranked list, where the user might stop reading at any time).

The tweet timeline generation task supplements the standard challenges of ad hoc retrieval with issues from topic detection and tracking (TDT) and multi-document summarization. In our conception, systems need to address two additional challenges (beyond ad hoc retrieval):

- Detect (and eliminate) redundant tweets. This is equivalent to saying that systems must detect novelty.
- Determine how many results to return. Some topics have more relevant and non-redundant tweets than others and a system must be able to automatically infer this. Systems can make different precision/recall tradeoffs along these lines.

Redundancy is operationalized as follows: for every pair of tweets, if the chronologically later tweet contains substantive information that is not present in the earlier tweet, the later tweet is considered novel; otherwise the later tweet is redundant with respect to the earlier one. In our definition, redundancy and novelty are antonyms, and so we use them interchangeably, but in opposite contexts.

Note that because of the temporal constraint, redundancy is not symmetric. If tweet A precedes tweet B and tweet B contains substantively similar information found in tweet A, then B is redundant with respect to A, but not the other way around. Finally, we assume transitivity. Suppose A precedes B and B precedes C: if B is redundant with respect to A and C is redundant with respect to B, then by definition C is redundant with respect to A. In this task setup, redundancy boils down to the definition of the binary relation “contains substantively similar information”. This is more precisely defined as part of our evaluation methodology, which is described in Section 3.2.

We imagined that participants would tackle the TTG task in a pipelined architecture that begins with ad hoc retrieval followed by summary generation. Therefore, participants in the TTG task were also required to participate in the ad hoc retrieval task (i.e., submit runs).

The tweet timeline generation task represents a natural extension of classic ad hoc retrieval and shares similarities with previous tasks such as aspect retrieval [10], sub-topic retrieval [17], and the notion of “information nuggets” in TREC question answering evaluations [15, 3]. From the perspective of search result diversification, Tao et al. [13] recently explored multi-aspect retrieval in the tweet context. Previous work along these lines suggest that this task is not so difficult as to preclude meaningful progress toward its solution, which is an important consideration in developing TREC tasks.

3. EVALUATION METHODOLOGY

3.1 Temporally-Anchored Ad Hoc Retrieval

The ad hoc retrieval task was evaluated using a standard pooling methodology by NIST assessors. Judgment pools were created using depth 100 across all submitted ad hoc runs, plus a random selection of 100 tweets per topic from each TTG run. Although we envisioned a system architecture consisting of ad hoc retrieval followed by summary generation, the inclusion of TTG results into the pool was explicitly designed to reduce missing judgments for TTG runs in cases where participants’ systems did not include an explicit ad hoc retrieval stage.

These judgment pools were further reduced by removing retweets (declared not relevant by track fiat) and then clustered so that textually similar tweets are presented to assessors in close proximity (to enhance judgment consistency). Tweets were judged on a three-way scale of “not relevant”, “relevant”, and “highly relevant”.

3.2 Tweet Timeline Generation

The TTG definition of redundancy and the assumption of transitivity means that the task can be viewed as semantic clustering—that is, we wish to group relevant tweets into clusters in which all tweets share substantively similar information. Within each cluster, the earliest tweet is novel; all other tweets in the cluster are redundant with respect to all earlier tweets.

Our annotation methodology to generate judgments for evaluation builds on exactly this idea. We begin with a list of all relevant tweets, ordered chronologically, from earliest to latest. These tweets are presented, one at a time, to a human assessor. For each tweet, the assessor can add it to a previously-existing cluster if she thinks the tweet contains substantively similar information with respect to tweets in the existing cluster, or she can create a new cluster for the tweet. We have developed a JavaScript-based annotation interface to help assessors accomplish this task—a screenshot is shown in Figure 1.

In the interface, the next tweet to be processed is shown at the bottom of the screen. The assessor can either add the tweet to an existing cluster by clicking the “Add” button next to the cluster or create a new cluster by hitting the space bar. At any time, the assessor can expand a cluster to show all tweets contained in it, or collapse the cluster, in which only the first tweet is shown. The interface also implements an undo feature that allows the assessor to reverse the action taken and go back to the previous tweet.

The TTG evaluation methodology boils down to this central question: what exactly does “substantively similar information” mean? Like document relevance in ad hoc retrieval, assessors make the final determination and we expect natural variations among humans. However, pilot studies helped us devise a set of guidelines, which were provided as instructions to the assessors. We told them: A good rule of thumb
Figure 1: Screenshot of the clustering interface for assessors. Tweets are presented one at a time in chronological order. The next tweet to be processed is shown at the bottom of the screen: the assessor can either add the tweet to an existing cluster (by clicking the “Add” button next to the cluster) or create a new cluster (by hitting the space bar). At any time, the assessor can expand a cluster to show all tweets contained in it, or collapse the cluster, in which only the first tweet is shown.

is that if two tweets “say the same thing”, then they’re substantively similar.

To provide further guidance, we devised a number of questions that the assessor might consider in determining whether two tweets should be in the same cluster:

- If I had already seen the first tweet, would I have missed out on some information if I didn’t see the second tweet?

- If two tweets are similar but the second contains an addition to or endorsement of the first, is the addition/endorsement important enough that I would be interested in seeing both tweets?

- Sometimes two tweets look similar but actually narrate the development of an event. Are the tweets different enough from each other that I would want to see both tweets to understand how an event develops or unfolds?

The annotation process proceeded concurrently at the University of Maryland and the University of Illinois, using relevant documents from the NIST judgment pools as the starting point. The two assessors at the University of Maryland were graduate students in computer science (both male). The two assessors at the University of Illinois were graduate students in library and information science (one male, one female). Assessors were first trained in the laboratory: the session included an introduction to the task and an overview of the annotation interface. After that, assessors were free to perform annotations at their own pace on their laptops, at locations of their choosing (this was possible because the annotation interface was implemented in JavaScript and hence accessible over the web). All assessors began with a throwaway “practice topic” (although they were not aware of the throwaway nature) and then proceeded to annotate topics in batches (roughly ten topics per batch). Topics were grouped into batches of roughly equal size (in terms of the number of relevant documents). When an assessor completed a batch, he or she could request another batch to work on. In order to preserve consistency across topics, we opted to have fewer annotators each working on more topics, as opposed to many annotators each processing only a single batch.

It is our intention to produce two independent sets of cluster annotations over all topics (one set from each site) so that we can study annotator differences and their effects on evaluation stability. At the time of this writing, we have a “covering set” over all topics by combining results from both sites, although we do not have two sets of annotations for every topic yet.

The output of the human annotation process is an ordered list of tweet clusters. Within each cluster, the tweets are sorted by temporal order (earliest to latest). The clusters themselves are sorted by the temporal order of their earliest tweet. Following the heuristic of using the most straightforward metric when defining a new task (and then subsequently refining the metric as needed), we decided to measure cluster-based precision and recall. The measure is cluster-based in the sense that systems only receive credit for returning one tweet from each cluster—that is, once a tweet is retrieved, all other tweets in the cluster are automatically considered not relevant. From this, we can compute precision, recall, and F-score in the usual way (lacking any basis for setting the $\beta$ parameter, we simply computed $F_1$). Since the user model assumes that a searcher will consume the
The entire summary, set-based metrics seemed appropriate and straightforward.

The only additional refinement is that we computed both weighted and unweighted variants of recall. In weighted recall, each cluster is assigned a weight proportional to the sum of relevance grades from every tweet in the cluster (relevant tweets receive a weight of one and highly-relevant tweets receive a weight of two). This weighting scheme implements the heuristic that larger clusters and those containing more highly-relevant tweets are more important, and the denominator in the weighted recall computation is the sum of clusters’ weights. In unweighted recall, all clusters are considered equally important, and the denominator is simply the total number of clusters.

Note that this setup gives equal credit to retrieving any tweet from a cluster. Intuitively, however, this seems overly simplistic—users would certainly prefer seeing certain tweets over others, even if they contain substantively similar information. For example, users might prefer to see the earliest tweet, a tweet from the most “authoritative” user (e.g., a verified news account), or a tweet from someone close by in their network (e.g., a tweet from someone they follow). We currently do not have sufficient understanding to accurately model such preferences, and thus explicitly made the decision not to tackle this challenge.

The evaluation metrics for TTG were derived from previous work referenced in Section 2.2: aspect recall [10], subtopic recall [17], and the “nugget pyramid” approach from the TREC question answering evaluations [6]. Alternative metrics we had considered include those based on gain [2] and the extension of mean average precision to graded relevance judgments [11]. After careful consideration, for this initial evaluation we decided to stick with the simpler set-based metrics, but we will consider different metrics in the future based on lessons learned.

4. RESULTS AND DISCUSSION

4.1 Temporally-Anchored Ad Hoc Retrieval

For the temporally-anchored ad hoc retrieval task, NIST received a total of 75 runs from 21 groups. Table 1 shows the run with the highest mean average precision (MAP) from each group. Precision at rank 30 (P30) and R-precision are also shown. Rows in the table are sorted by MAP. In computing these metrics, both “relevant” and “highly relevant” grades are considered relevant.

For reference, we provided two baselines, also shown in Table 1: The “baseline” condition is simply the raw output of the API with queries as bags of words. The “RM3” condition is our reference implementation of the RM3 variant of relevance models [5, 1]. For RM3, we extracted the top 20 feedback terms from the top 50 tweets, which is interpolated with the original query model with a weight of 0.5.

These results show that participants are submitting highly-effective runs overall. Using the baselines to calibrate, there are many more teams beating the baselines than in previous years. This suggests that the ad hoc retrieval task is perhaps becoming “too easy”.

4.2 Tweet Timeline Generation

In total, 13 groups submitted 50 runs to the tweet timeline generation task. Each topic averaged 194 relevant documents. In the reference clusters, assessors formed an average of 87 clusters per topic (and each cluster averaged 2.2 tweets). Results based on these annotations are shown in Table 2, which contains the run with the highest weighted $F_1$ score from each group. The columns show unweighted recall, weighted recall (indicated by the $^w$ superscript), precision, $F_1$ with unweighted recall, and $F_1$ with weighted recall (indicated by the $^{w}$ superscript). Rows are sorted by weighted $F_1$ score.

Recognizing that systems make different choices with respect to balancing precision and recall, it is illustrative to visualize the tradeoffs in a scatter plot. Figure 2 shows precision vs. unweighted recall (left) and precision vs. weighted recall (right) for all runs. Iso-$F_1$ contours are plotted in blue; points on the same contour line have the same $F_1$ score, but with different precision/recall tradeoffs.

What is the effect of the cluster weights? Figure 3 shows a scatter plot of the weighted $F_1$ score vs. the unweighted $F_1$ score for all submitted runs (where each point represents a run). We see that for most runs, the two scores

1https://github.com/lintool/twitter-tools/tree/master/twitter-tools-rm3
Run | Group | MAP | P30 | R-prec
--- | --- | ---- | ---- | ----
PKUIICAST3 | PKUIICAST | 0.5863 | 0.7224 | 0.5727
hltcoe3 | hltcoe | 0.5707 | 0.7121 | 0.5660
ECNURankLib | ECNUCS | 0.5529 | 0.7133 | 0.5427
PolyURun1 | POLYUCOMP | 0.5402 | 0.6994 | 0.5468
ICARUN2 | ecnu | 0.5327 | 0.6909 | 0.5435
QUQueryExp5D25T | QU | 0.5155 | 0.6697 | 0.5158
HPRF1020RR | QCRI | 0.5122 | 0.6982 | 0.5086
Pris2014a | BUPT_PRIS | 0.5005 | 0.7012 | 0.4995
NovaRun2 | NovaSearch | 0.4873 | 0.6709 | 0.4950
UCASRun3 | UCAS | 0.4703 | 0.6697 | 0.4774
SCIAI14a | SCIAITeam | 0.4407 | 0.6503 | 0.4707
ERLU | ir.cs.sfsu | 0.4200 | 0.6291 | 0.4493
UDInfoQE | udel_fang | 0.4154 | 0.6115 | 0.4414
ICTNETRUN4 | ICTNET | 0.4141 | 0.6248 | 0.4369
UWMHBUT2 | UWM.HBUT | 0.3427 | 0.5121 | 0.4055
RM3 | baseline | 0.3394 | 0.5176 | 0.3963
JufeLdkeAdhoc1 | LDKE | 0.3090 | 0.5145 | 0.3793
NewBee | zhg15 | 0.2745 | 0.4485 | 0.3176
udelRunAH | udel | 0.1841 | 0.5103 | 0.2485
wistudt2bd | wistud | 0.1730 | 0.3018 | 0.2128
OSIM | BJUT | 0.1708 | 0.3297 | 0.2207
SRTLAH | HU_DB | 0.0505 | 0.2527 | 0.0561

Table 1: Results of the temporally-anchored ad hoc retrieval task, showing the run with the highest mean average precision (MAP) from each group. Precision at rank 30 (P30) and R-precision are also shown. Rows are sorted by MAP.

Run | Group | recall | recall | precision | F1 | F1w
--- | --- | ---- | ---- | -------- | --- | ----
TTGPKUIICAST2 | PKUIICAST | 0.3698 | 0.5840 | 0.4571 | 0.4088 | 0.5128
EM50 | QCRI | 0.2867 | 0.4779 | 0.4150 | 0.3391 | 0.4442
hltcoeTTG1 | hltcoe | 0.4029 | 0.5915 | 0.3407 | 0.3692 | 0.4324
QUTmpDecayTTgCL | QU | 0.3277 | 0.5167 | 0.3236 | 0.3256 | 0.3980
SRTL | HU_DB | 0.0942 | 0.2851 | 0.5615 | 0.1613 | 0.3782
PrisTTG2014b | BUPT_PRIS | 0.4231 | 0.6137 | 0.2730 | 0.3319 | 0.3779
3unique0 | uog_twteam | 0.2522 | 0.4374 | 0.2558 | 0.2540 | 0.3228
UDInfoMMRWC5 | udel_fang | 0.0900 | 0.2191 | 0.5709 | 0.1555 | 0.3167
udelRunTTG1 | udel | 0.1873 | 0.3645 | 0.2793 | 0.2242 | 0.3163
wistudt2bd | wistud | 0.1111 | 0.3075 | 0.2827 | 0.1595 | 0.2946
SCIAI3cm4aTTG | SCIAITeam | 0.0655 | 0.1941 | 0.4992 | 0.1158 | 0.2795
ICTNETAP3 | ICTNET | 0.2234 | 0.4623 | 0.1792 | 0.1989 | 0.2583
JufeLdkeSum2 | LDKE | 0.4294 | 0.6156 | 0.0861 | 0.1434 | 0.1511

Table 2: Results of the tweet timeline generation (TTG) task, showing the run with the highest weighted F1 score from each group. Columns show recall (unweighted and weighted), precision, and F1 (unweighted and weighted); the superscript indicates the weighted variant of the metric. Rows are sorted by weighted F1 score.
are highly correlated, but there are a number of runs where the weighted $F_1$ score is much higher than the unweighted $F_1$ score. This is an interesting observation that warrants additional analysis.

To verify the stability and reliability of the evaluation, we performed two sets of analyses: the first concerns the number of missing judgments in evaluating the TTG runs. Recall that our cluster annotation workflow starts with relevant tweets from the NIST judgment pools, created via the process described in Section 3.1. In total, the TTG runs returned a combined 81,726 unique tweets. Of these, 37,449 (45.8%) were absent from the judgment pools and therefore lacked explicit judgments. However, these missing tweets were concentrated in a relatively small number of runs: this is shown in Figure 4, which plots the fraction of tweets that are missing judgments (averaged across topics). For 18 runs (out of 50), we did not observe any missing judgments. Of the runs with missing judgments, 19 were missing less than 5%. Note that these bars indicate the fraction of missing judgments, which obscures the absolute number since some systems return answers that are on average longer. For example, the run that was most severely impacted had a average of 411 missing judgments per topic (out of an average of 1000 tweets returned).

There are two possible interpretations of these results: the first is that systems are building summaries without an explicit ad hoc retrieval stage (whose outputs can contribute to the ad hoc judgment pools); the second is that the systems are generating summaries that are so long that missing judgments are the result of a shallow pool depth. The TTG runs with the largest number of unjudged results appear to have been generated without a corresponding ad hoc run. These runs are also very verbose for the most part. On the whole, it is the case that the number of missing judgments is positively correlated with run length, but the fraction of results that are unjudged does not show this relationship. From this, there appears to be evidence in favor of the hypothesis that the shallow pool depth is the reason for missing judgments. However, without explicit knowledge of each team’s operational relationship between the ad hoc and TTG runs, it is hard to draw firmer conclusions.

Our second analysis focuses on the stability of the evaluation with respect to assessor differences. The judgment of whether two tweets contain “substantively similar information” is likely to have high variability across assessors, which would yield substantially different clusters. We would like to determine to what extent this impacts our ability to make system comparisons, i.e., that system $X$ is more effective than system $Y$ [14]. Note that as is standard in IR meta-evaluation, differences in absolute scores are not worrisome, so long as system comparisons are stable.

At the time of this writing, we have 11 topics that contain two independent sets of cluster annotations. Figure 5 shows scores based on the “official” judgments and the alternate judgments for each of the five metrics in Table 2. Results are sorted by scores based on the official judgments. The official set averaged 73 clusters per topic, while the alternate judgments averaged 59 clusters per topic, which indicates that humans perform the clustering task at different levels of granularity. We see that the rankings produced by both sets of judgments are highly correlated, with the exception of weighted $F_1$, which shows a number of runs that yield different comparisons with respect to the two sets of judgments. Furthermore, except for the weighted variants, the absolute values of the metrics are also quite similar. We are currently looking into the causes of the behavior of the weighted metrics.

An alternative view of these analyses is shown in Table 3, where we tally the number of rank swaps for each metric. A rank swap is a pairwise comparison where according to one set of judgments, run $A$ scores higher than run $B$, but according to another set of judgments, run $B$ scores higher than run $A$. Note that there are a total of $(50 \times 49) / 2 = 1225$ pairwise comparisons. Table 3 also shows the Kendall’s $\tau$ correlation between rankings induced by the two different sets of judgments. Note that with the exception of weighted $F_1$, we do not observe many rank swaps, as confirmed by the high rank correlations.

Finally, we show a histogram of the rank swaps for weighted $F_1$ in Figure 6 binned by absolute score differences. Such an analysis is informative because we are less concerned with rank swaps in which the absolute score differences between...
Figure 5: Comparison between scores based on the official judgments and the alternate judgments for unweighted recall, weighted recall, precision, unweighted $F_1$, and weighted $F_1$. Runs are sorted by score based on official judgments in descending order.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Rank Swaps</th>
<th>Kendall’s τ</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted recall</td>
<td>31</td>
<td>0.948</td>
</tr>
<tr>
<td>weighted recall</td>
<td>58</td>
<td>0.905</td>
</tr>
<tr>
<td>precision</td>
<td>38</td>
<td>0.936</td>
</tr>
<tr>
<td>unweighted $F_1$</td>
<td>37</td>
<td>0.938</td>
</tr>
<tr>
<td>weighted $F_1$</td>
<td>147</td>
<td>0.760</td>
</tr>
</tbody>
</table>

Table 3: Number of rank swaps and overall Kendall’s $\tau$ correlation based evaluation with the two sets of judgments for each scoring metric.

Figure 6: Histogram of rank swaps for weighted $F_1$ binned by absolute score differences.

5. CONCLUSION

The tweet timeline generation task this year represents a serious attempt in the TREC Microblog track to move beyond ad hoc retrieval. In the design of the track, we have explicitly taken a conservative approach in changing only one aspect of the evaluation at a time (last year, it was the introduction of the evaluation-as-a-service model; this year, the introduction of TTG). We believe that this approach provides continuity and allows participants to build on lessons learned in previous years in an incremental fashion. The track will continue in TREC 2015, and we look forward to discussions about future developments with participants at the TREC workshop.

6. ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under IIS-1217279 and IIS-1218043. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor.
7. REFERENCES


