# New Techniques for Coding Political Events Across Languages

Yan Liang, Khaled Jabr, Christan Grant
School of Computer Science
University of Oklahoma
Norman, Oklahoma, 73019
{yliang, khaled.jabr-1, cgrant}@ou.edu

Jill A. Irvine
International and Area Studies
University of Oklahoma
jill.irvine@ou.edu

Andrew Halterman
Political Science
Massachusetts Institute of Technology
ahalt@mit.edu

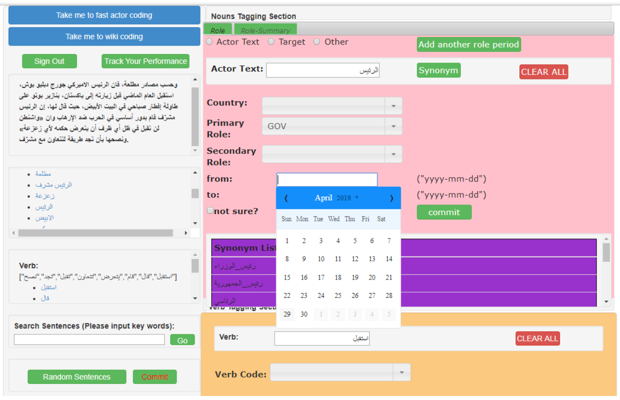
***Abstract*—Data produced from text is one of the most important new resources for doing research in quantitative political science research. "Event data," which codes structured actor-event-target triples from text, is a particularly useful form of data. Most publicly available event datasets, though, are limited to English only, limiting their usefulness for studying many regions. We demonstrate new techniques for coding events in English, as well as in Arabic, a previously uncoded language. In order to generate language-specific political event data, "actor" and "verb" dictionaries are required for each specific language. Efficiently developing an accurate and extensive dictionaries is a difficult challenge. In this paper, we describe four different approaches we have used to solve the problem of producing dictionaries and how other researchers can use our ideas to develop dictionaries in a new language or new ontology. This work stems from an ongoing NSF RIDIR project, "Modernizing Political Event Data" which aims to produce multilingual event data and the software needed for researchers to produce custom datasets.**

## I. Introduction

Data produced from news stories is one of the most important new sources of information for quantitative political science research. Drawn from English language news reporting around the world, machine coded data of geo-referenced political and social activity has opened up new possibilities for the study of numerous political phenomena ranging from social movements to violent conflict and unrest and government responses. Data is automatically coded from text by comparing phrases in the text to hand coded dictionaries to identify actors and events. However, one limitation of event data is its restriction to English language sources only. We aim to build a new machine-coded event data set of Arabic news corpus and detect events such as 'protest' or 'attack' from the news corpus using the event coder UniversalPetrarch[1]. While UniversalPetrach's code only requires minor changes to accommodate new languages, the dictionaries used to map phrases to codes need to be completely re-written for each new language. Each dictionary encodes several specific cases used to translate sentences to events. Several issues make automated event coding in multiple languages a difficult and an as yet unsolved task, including the unwieldy size of text data which can reach terabytes, extracting the relevant actors to the events being studied, and guaranteeing both the accuracy and abundance of the coded results.

[1]UniversalPetrarch is a language-agnostic political event coding system https://github.com/openeventdata/UniversalPetrarch/

This paper describes our process of tackling these issues and building a computer-assisted coding system for Arabic dictionary development. While the dictionaries we produce are Arabic-specific, the tools we develop will work for researchers making dictionaries in any language.



وحسب مصادر مطلعة، فان الرئيس الاميركي جورج دبليو بوش، استقبل العام الماضي قبل زيارته إلى باكستان، بنازير بوتو على طاولة إفطار صباحي في البيت الأبيض، حيث قال لها، إن الرئيس مشرّف قام بدور أساسي في الحرب ضد الإرهاب وان «واشنطن لن تقبل في ظل أي ظرف أن يتعرض حكمه لأي زعزعة» ونصحها بأن تجد طريقة للتعاون مع مشرّف.

- مطلعة
- الرئيس مشرف
- زعزعة
- الرئيس
- الابيض

Verb:
["استقبل","قال","قام","يتعرض","التعاون","تقبل","تجد","نصح"]
- استقبل
- قال

Fig. 1. Regular coding interface, Extracted Noun and Verb sentences are on the left, clicking populates the actor on the right.

## II. Background: Creating Political Event Data

In the past 15 years, automated methods of text analysis have become prominent in political science research, allowing researchers to study much larger amounts of text data, in order to find meaning that is difficult to extract manually. Event data in political science consists of a "triple" of basic information: an event such as an attack or protest, performed by a source actor, against a target. These events, actors, and targets are automatically recognized in text, extracted, and resolved to a defined set of ontology codes, such that "demonstrated" and "rallied in the streets" would both be coded as a PROTEST event and "Angela Merkel" and "German Ministry of Defense" would both be represented as DEU GOV in the ontology we use [1]. Performing this process on many millions of documents produces a set of structured data that is much easier to analyze than the raw documents [2], [3].

The full process of producing events from text consists of three steps. First, the text is put through a natural language processing step which annotates the sentences with grammatical information about nouns and verbs. For our work, we use CoreNLP pipeline [4] to achieve this [2]. The second step is to find potential events in the text. Based on the grammatical structure we produced in step 1, we feed our data into UniversalPetrarch, which looks for combinations of noun and verb phrases that are likely to be actors, targets and actions in an event. A limitation in this process is that UniversalPetrarch does not consider the content of the data at this point but only the grammatical structure of the sentence, which can cause it to extract events that are not interesting or relevant, such as sports stories or marriage announcements. This will be addressed when we filter our data and only deliver useful events to our coders.

The third step is also performed by UniversalPetrarch and consists of comparing the extracted actor and action text to a defined set of phrases and the codes that should be assigned to them. The last step is crucial for the event data analysis, since even after events have been recognized and extracted from text, the sheer variety of terms and languages to refer to people, organizations, and events means that raw text phrases are impossible to analyze quantitatively on their own. Resolving them to common codes makes further analysis feasible. For example, if the previous step recognizes "marched and chanted slogans" as an action in the fragment of text, this step would resolve it to a consistent, defined event type, such as PROTEST. Similarly, the extracted actor text "Angela Merkel" could be resolved to DEU GOV in the CAMEO ontology we use.

## III. Related Work

Philip Schrodt and colleagues at the University of Kansas created the original English language event coding dictionaries. To develop the dictionaries, TABARI [2], the coding program that Schrodt used displayed sentences to coders if the system recognized an event in a sentence but the actor was not in the dictionary. Coders were then responsible for adding these new phrases to the dictionaries.

Our process improves TABARI's approach in several ways. First, it is a web-based system, allowing many coders to work together without interfering with each others' work. Second, we provide useful structure to the task by suggesting the CAMEO [5] ontology in a dropdown list instead of asking coders to either memorize or refer to the codebook for the list of possible codes. We used other technologies to facilitate our process, for example, using word2vec [6] to suggest synonyms of relevant actors to be added to the dictionary. We also developed a validation system based on peer review allowing coders to flag the dictionary entries about which they are not confident. All of these tools improve our speed and accuracy in generating dictionaries.

Javier Osorio and colleagues worked on dictionary development for Spanish [7], but because their text was drawn from politically relevant sources, they did not need to develop mechanics to filter out less useful text from a large text corpus as we needed to do. Because we used a large corpus of data, a challenge for our coders was the number of stories they received that had no relevance for coding political events. In order to address this problem, we developed a way to filter out stories of no clear relevance, delivering stories to coders most likely to contain political events. Another aspect of coding we took into consideration in Arabic dictionary development is an actor's role during different time periods since each actor might serve different roles at different times. That information is important when detecting new political events and we built systems to facilitate the process of recording that piece of information. An additional innovation we developed was to automatically find Arabic transliterations for existing actors in the English dictionary using Wikipedia. Our "wiki-bio" coding approach pre-populates all the possible roles an actor might occupy in different time ranges with a Wikipedia link attached and is much more convenient for coders to use.

Martin Atkinson *et al.* worked on extracting security-related events corpus from a multi-lingual corpus but did not build a system for a specific language like our system does for Arabic [8]. The way the authors cluster events is SVM-based [9], while we use a dictionary-based system, UniversalPetrarch, that uses dictionary entries to map phrases to a specific CAMEO code. This dictionary-based system should be able to capture more specific events of relevance than machine-learning systems and achieve higher accuracy, but it requires a lot of work to build a language-specific dictionary. Rapid development of dictionaries is therefore needed in order to make dictionary-based event coding feasible for new languages and domains.

## IV. Coding Interfaces

Arabic-language actor dictionary development is crucial for the event detection step (3rd step) performed by UniversalPetrarch. We used different techniques, ranging from manual to fully automatic, to help our team of human annotators (described in the next section) create the Arabic-language actor dictionaries. In this section, we outline each approach, discuss its advantages and limitations, and describe the interface we built for each technique. Finally we make recommendations to other researchers creating non-English language actor dictionaries or researchers developing dictionaries using new ontologies.

The first approach we developed was to sample the data from our Arabic Gigaword text corpus and use CoreNLP ( [4]) to parse the data into a grammatical format with nouns and verbs and feed the parsed result to UniversalPetrach [2]. The next step was for our coders to code an entry for each extracted actor produced by UniversalPetrarch. To do so, we developed our main coding interface, Interface 1 (Figure 1), where we used several new techniques for this step in order to enhance accuracy and coding speed.

First, we exposed a search option for the coders with a "text" index on our data in MongoDB, so that coders can

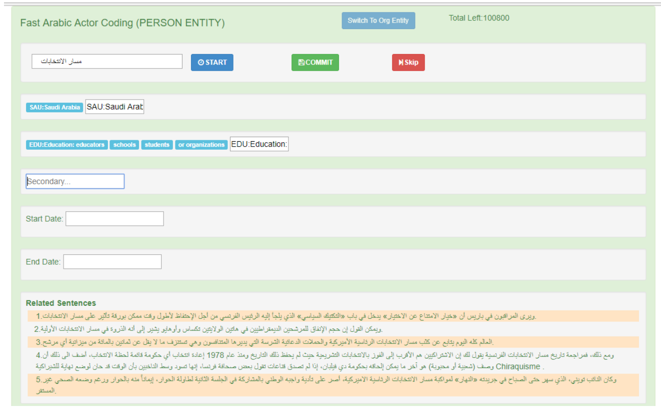
[2]http://eventdata.parusanalytics.com/software.dir/tabari.html/

work on related topics by querying similar keywords. It also allowed them to focus on coding similar topics, speeding up the process. Second, when an actor entry is already coded, and the same word appears in the current sentence, its coded record from the database will pop up on the interface. Other coders can then confirm the previous coder's work, rather than entering a new entry. This set-up ensures that the most common entries are reviewed most frequently. Third, the interface will also suggest alternative spellings of the names with its "synonym" based on word2vec. This greatly increases the yield of the system, since many versions of the same name can be added once to the dictionary. Finally, the interface also includes an "unsure" button which flags the entry for peer review or review by the supervisor of the coding process. More on this is in the experiments section.

The advantage of this interfaces and the techniques it employs is that it allows for broad coverage of the text by pulling out all the possible actors in the text and ensures that the dictionaries will have entries for them. The disadvantage is clear: we are randomly sampling sentences out of a corpus of millions of sentences, with the result that the coders might code actors who are not, in fact, high priority for adding. In order to solve this problem, we applied a topic modeling strategy, latent dirichlet allocation [10] to our sentences so we can choose the "politically relevant" topics to code. We clustered the sentences into different number of topics ($N \in \{5, 10, 20, 40\}$), We sampled sentences from each cluster to show different native Arabic speaking coders and asked them to summarize which topic each clustering is mainly about. We then used their assessment to conclude that clustering the corpus into 20 groups gives us the best results. Finally, we filtered out unrelated topics like sports from the corpus, only showing the sentences that come from more "political related" topics. Using this technique and the interface built for it, the coders added 6,387 actor entries and 1,628 verb entries.

The second approach we used in actor coding was to directly translate the existing English actor dictionary to Arabic using Wikipedia. Because each record is costly to add, being able to translate existing English dictionary entries into Arabic would greatly increase our efficiency. For each actor in the English dictionaries, we attempted to find its Wikipedia page by an exact name match. Once we located an English article, we then checked to see if a corresponding Arabic article existed. If it did, we took the Arabic name for this actor and the existing role information in the English language dictionary and added them to the Arabic dictionaries. The major advantage of this approach is its efficiency and speed, as no human effort is needed. The limitation of this approach is that we can only add actors that are already in the existing English dictionaries. Additionally, entries may not include each role of the relevant actors in Arabic language news sources. With this approach, we were able to achieve 5,696 actor entries. We did not develop a special interface.

The third approach we used in actor coding was to automatically find high-frequency actors in the corpus and to have coders create entries for them. We first use LDA to filter out unrelated topics, and then pass the remaining documents through CoreNLP to parse the sentences into noun- and verb phrases. We then used a MapReduce [11] strategy to count the frequency of each actor in the corpus and rank the count in decreasing order. Next, we filter out all nouns that are not named entities using a multilingual named entity extraction (NER) model [12]. We then supply the actors to the coders ranked by decreasing order of appearance. Along with the actor or organization showing on the interface, we also presented five sentences in the corpus where the noun appears as background content to our coders, so the coders could have more content-related information when they made a coding decision. A screen shot of the interface is shown in Figure 2.



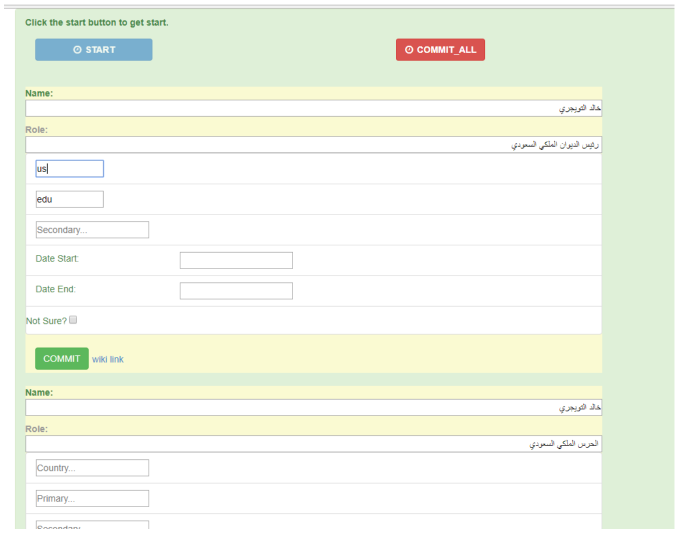
Related Sentences

1. ويرى المراقبون في باريس أن «خيار الامتناع عن الاختيار» يدخل في باب «التكتيك السياسي» الذي يلجأ إليه الرئيس الفرنسي من أجل الإحتفاظ لأطول وقت ممكن بورقة تأثير على مسار الانتخابات.
2. ويمكن القول إن حجم الإنفاق للمرشحين الديمقراطيين في هاتين الولايتين تكساس وأوهايو يشير إلى أنه الذروة في مسار الانتخابات الأولية.
3. العالم كله اليوم يتابع عن كثب مسار الانتخابات الرئاسية الأميركية والحملات الدعائية الشرسة التي يديرها المتنافسون وهي تستنزف ما لا يقل عن ثمانين بالمائة من ميزانية أي مرشح.
4. ومع ذلك، فمراجعة تاريخ مسار الانتخابات الفرنسية يقول لك إن الاشتراكيين هم الأقرب إلى الفوز بالانتخابات التشريعية حيث لم يحفظ ذلك التاريخ ومنذ عام 1978 إعادة انتخاب أي حكومة قائمة لحظة الانتخاب، أضف الى ذلك أن وصف (شعبية أو محبوبة) هو آخر ما يمكن إلحاقه بحكومة دي فيلبان، إذا لم تصدق قناعات تقول بعض صحافة فرنسا، إنها تسود وسط الناخبين بأن الوقت قد حان لوضع نهاية للشيراكية Chiraquisme .
5. وكان النائب تويني، الذي سهر حتى الصباح في جريدته «النهار» لمواكبة مسار الانتخابات الرئاسية الاميركية، أصر على تأدية واجبه الوطني بالمشاركة في الجلسة الثانية لطاولة الحوار، إيماناً منه بالحوار ورغم وضعه الصحي غير المستقر.

Fig. 2. Fast coding interface, Frequency and NER based, 5 sentences containing the keyword is showed on the bottom

The advantage of this method is that it recommends the most important actors (measured by frequency) in the corpus and directs the coders to work on them. Unfortunately, several off-the-shelf programs exhibited poor performance when performing Arabic NER. The NER model trained in spaCy with poor data, so its performance is inadequate in recognizing person and organization names. It also cannot distinguish between politically relevant and irrelevant people. Out of the most frequent 7,180 people or organizations recognized by the spaCy multilingual model, only 204 were political actors that could be added to the dictionary. This low yield is attributable to the system returning text that is not a named entity, is political irrelevant or names that are too vague (i.e common first or last names). The disadvantage of this approach is substantial: coders spend a large amount of time skipping irrelevant actors in order to find one to add to the dictionary, which is frustrating and time-consuming. In order to enhance the performance of this approach, we need to build a customized Arabic-specific NER system. To build this system, our coders have annotated 6,000 Arabic sentences for NER model training and are currently actively working on them. When that task is completed, we can train an improved NER model, which will greatly improve the yield of this approach.

The fourth approach we used was leveraging the information available on Wikipedia. Our target Wikipedia data was the data in the info box of politically relevant actors. To scrape the data, we compiled a list of links of Wikipedia categories, such as categories of government ministers, and prominent politicians in the 22 Arab states, and wrote a specialized scraper to extract this Wikipedia data. The data contained the name of the actor, any roles he/she occupied, and any dates associated with these roles. Once we scraped the data, we developed our third web-based interface, "Wiki-Interface" (Figure 3), in which the actors' data was repopulated into the interface, and presented to the coders. The coders insured the consistency of the data, such as removing any extra title information from the actor's name, and translating names and roles from English to Arabic for actors with no Arabic Wikipedia pages. Coders also discarded actor entries that had no role data associated with them. Each actor role was presented to the coders as a card which they could either commit or discard. A link to the Wikipedia page that the current actor data was scraped from was also provided to help coders disambiguate any uncertainties they had. Using this interface we generated entries for 2,327 actors, totaling 4,286 role period ranges.


Click the start button to get start.

Name: خالد التويجري

Role: رئيس الديوان الملكي السعودي

us

edu

Secondary...

Date Start:

Date End:

Not Sure? ☐

Name: خالد التويجري

Role: الحرس الملكي السعودي

Country...

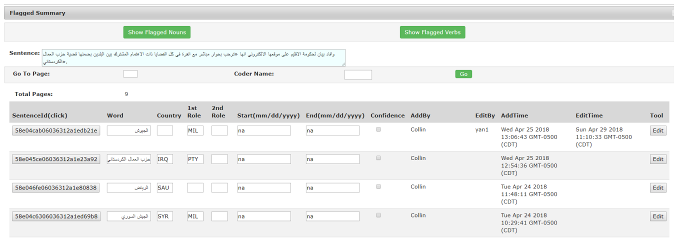
Primary...

Secondary...

Fig. 3. Wiki biography based interface, each "card" represent a role for the actor

The advantage of this approach is that coders are only working on the actors that are politically relevant since we only scraped entries from politically relevant Wikipedia category pages. Coding is also extremely fast, as the highly structured information is presented to the coders with the date ranges pre-populated. The disadvantage is that not all politically relevant actors have Wikipedia pages, nor do these pages always have biographical sidebars. Organizations also do not have biographical sidebars as people do, making this interface useful only for coding people.

With these aforementioned interfaces, and in order to get more accurate records, we implemented an "unsure" strategy, allowing coders to flag a record as unsure if coders were not confident about what they tagged. We then implemented a peer review interface for the coders to check each others' unsure records and make corrections, and had a supervisor track each coder's performance. In all our approaches, we kept our coders in the loop by taking their feedback as an input to our design process.


Flagged Summary

Sentence: وافاد بيان لحكومة الاقليم على موقعها الالكتروني انها «ترحب بحوار مباشر مع انقرة في كل القضايا ذات الاهتمام المشترك بين البلدين بضمنها قضية حزب العمال الكردستاني».

Go To Page: Coder Name:

Total Pages: 9

| SentenceId(click) | Word | Country | 1st Role | 2nd Role | Start(mm/dd/yyyy) | End(mm/dd/yyyy) | Confidence | AddBy | EditBy | AddTime | EditTime | Tool |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58e04cab06036312a1edb21e | الجيوش | | MIL | | na | na | ☐ | Collin | yan1 | Wed Apr 25 2018 13:06:43 GMT-0500 (CDT) | Sun Apr 29 2018 11:10:33 GMT-0500 (CDT) | Edit |
| 58e045ce06036312a1e23a92 | حزب العمال الكردستاني | IRQ | PTY | | na | na | ☐ | Collin | | Wed Apr 25 2018 12:54:36 GMT-0500 (CDT) | | Edit |
| 58e046fe06036312a1e80838 | الرياض | SAU | | | na | na | ☐ | Collin | | Tue Apr 24 2018 11:48:11 GMT-0500 (CDT) | | Edit |
| 58e04c6306036312a1ed69b8 | الجيش السوري | SYR | MIL | | na | na | ☐ | Collin | | Tue Apr 24 2018 10:29:41 GMT-0500 (CDT) | | Edit |

Fig. 4. Peer review interface to keep track of the unsure records and allow multiple people to view and verify.

## V. Coding Teams

In order to assist with our dictionary development, we hired 8-10 Arabic coders. The coders were mostly undergraduate students and native Arabic speakers with direct experience in teaching the language. We split the coders into two teams: Team 1 and Team 2. Within each team, coders were paired into groups of two to perform the task at hand and to verify it, with one performing and the second verifying. If both coders in the pair were unsure, the interfaces allowed the coders to flag the task. Other coders may then contribute to completing the task. Having such a team structure helped the coders get accustomed to the task faster and develop shared norms for approaching coding issues.

This team setup as well as our interfaces has many advantages. First, the division of the tasks and the verification mechanism ensured better results and higher accuracy. Second, our web-based interfaces allowed for much greater flexibility of work for our coders, especially whom were abroad or working remotely. The ease, clarity, and segmentation of tasks allowed for very cheap training cost, as well as fast adaptation of tasks by the coders. Third, having various interfaces allowed much more feedback and tweaking in the developmental side, which made it easier to accommodate coders needs.

The main disadvantage of our setup is the decentralization. Having various interfaces for related tasks can lead to unorganized and sparse results. To avoid that, we had a strict time-line to how we used those interfaces, which team used it, and when to transition to another interface, which seemed to help greatly in this regard.

Aside from using the interfaces to code, the coders attend weekly meetings with the interface development team to provide feedback about the interfaces and discuss questions about the coding process. Holding these meetings is an important aspect of eliciting feedback, understanding the language requirements, and assessing the coders' needs, which is a driving factor in the development of the most effective interfaces.

## VI. Evaluation

We track how much time each coder takes when working on the Wiki-Bio interface and the frequency-NER based interface

TABLE I
DESCRIPTION OF EACH INTERFACE USED BY CODERS FROM TEAM 1 AND TEAM 2.

| Interface | Teams | Description |
|---|---|---|
| Regular Interface (Figure 1) | Team1 | Extracted nouns and verbs from raw sentence with auto complete and synonym-aided feature. |
| Fast Coding (Figure 2) | Team1 | Frequency ordered and NER based PER and ORG coding. |
| Wiki Enhanced (Figure 3) | Team1, Team 2 | Provided pre-filled information from Wikipedia name cards. |
| NER Annotation (Figure 8) | Team1 | Interactive NER model updating interface provided by Prodigy. |

with the intention of understanding which interface works better for a given time budget. The coders' performance is presented in Table II below. We did not track time per task in the original interface because of the way we bundled many tasks into one: coders added between 0 and a half-dozen dictionary entries per sentence, with sometimes several roles for each actor, as well as identifying the source and target actors in the sentence. We therefore cannot unbundle the task timing to identify how long each discrete task took.

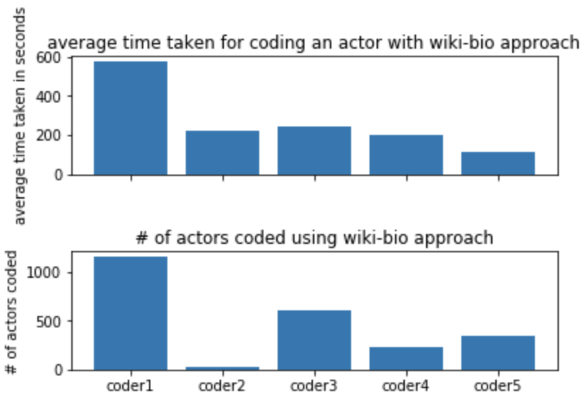


Fig. 5. Performance of wiki-based approach of 5 coders

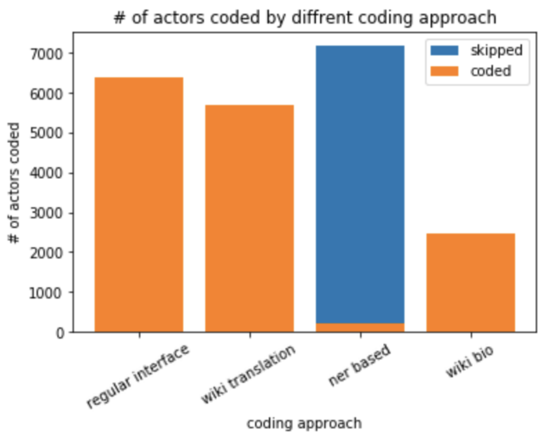


Fig. 6. Total number of actors coded for each approach

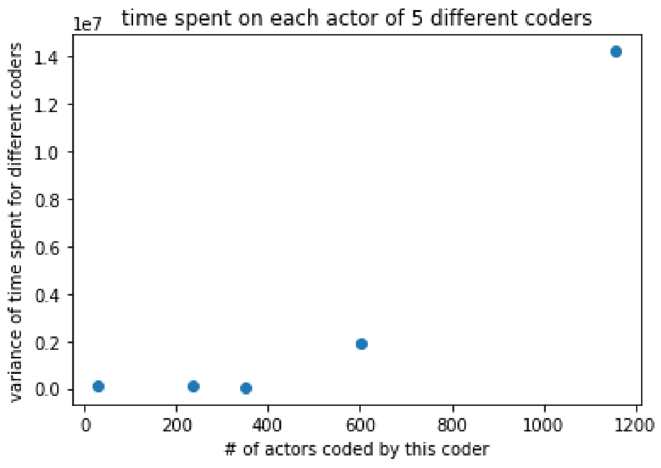


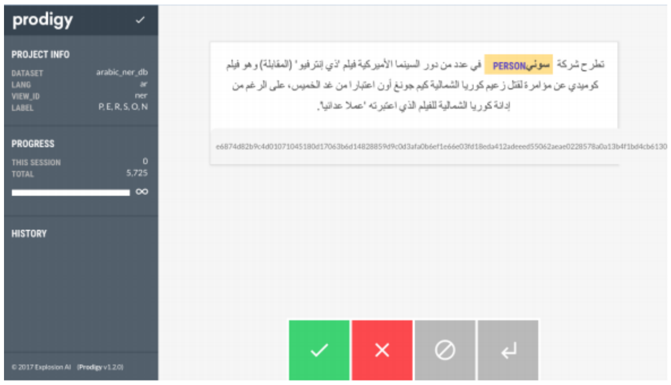
Fig. 7. Variance of time spent on each actor of different coders versus the number of actors coded by each coder

## VII. DISCUSSION

We know our NER based method was unlikely to perform well because the NER model we used was trained on poor multilingual data and does not work well on Arabic. Still, we were surprised at how poor the performance was: within 7,384 records we were only able to code 204 politically relevant ones. Once we find a political relevant entity with this interface, though, coding it only requires an average of one minute. One possible explanation for this relative speed is that we present five related sentences in which the actor appears so that the coders get a richer understanding of the actor and they can create the entry faster. This suggests to us that if we can get a better working version of Arabic-Language specific NER model, it will significantly enhance our yield and overall performance. We found the performance of Wiki-bio approach to be unexpectedly slow. We expected it to be faster than the NER-based coding approach since we had already pre-populated the time range for each entity and provided the URL to link the actor back to their Wikipedia page. For each actor entity it still took six minutes to code on average, and for each role it took roughly three minutes. A possible reason for the slow speed is that we do not provide the same background content in this interface as the NER based coding interface does. While coding different roles on each possible actor the coders still needed to review the code book to identify the correct role. Note that we only have 204 NER based actors coded, so the sample size is small and our time estimates may therefore be imprecise. Another interesting result we found related to coder efficiency in Fig.5: the longer a coder has been coding over time, and presumably the more experienced a coder becomes, the more average time it takes

TABLE II
PERFORMANCE OF CODERS WITH DIFFERENT CODING APPROACHES

| Approach | Actors coded | Actors Skipped | Total time (seconds) | seconds per Actor | Second per Role |
|---|---|---|---|---|---|
| Regular Interface | 6,387 | - | - | - | - |
| Wiki Translations | 5,696 | - | - | - | - |
| NER-based | 204 | 7,180 | 11,343 | 55.6 | - |
| Wiki bio | 2,459 | - | 926,289 | 377 | 202 |



تطرح شركة سوني PERSON في عدد من دور السينما الأميركية فيلم 'ذي إنترفيو' (المقابلة) وهو فيلم كوميدي عن مؤامرة لقتل زعيم كوريا الشمالية كيم جونغ أون اعتبارا من غد الخميس، على الرغم من إدانة كوريا الشمالية للفيلم الذي اعتبرته 'عملا عدائيا'.

e6874d82b9c4d01071045180d17063b6d14828859d9c0d3afa0b6ef1e66e03fd18eda412adeeed55062aeae0228578a0a13b4f1bd4cb6130

Fig. 8. Using Prodigy to train a named entity recognition system

the coder to code an actor. This may be because the more actors a coder encounters, the higher the probability that the coder will encounter ones that are more difficult to decide how to code. It could also be that as coders become more experienced, they are more likely to consider various possible roles or more complicated coding issues. From Fig.7 we can see that the variance of time per task increases along with the number of actors coded, which appears to support our first explanation.

## VIII. Summary and Related Work

Using the above approaches for developing dictionaries, we were able to complete Arabic actor and verb dictionaries with coverage equivalent to the English language dictionaries in less than two years of work, compared to the two decades that the English language dictionaries took to produce. We have used UniversalPetrach to generate events from our corpus of millions of Arabic sources using the dictionary we developed, and we expect to make comparisons between it and the English corpus after final debugging and quality checking. It is difficult to determine how many actor dictionary entries is sufficient for us to generate accurate event data. In any case, within our budget we aim to generate as many politically relevant actor and verb dictionary entries as possible, so we need the fastest possible coding framework to achieve this. We could potentially do better than one minute on NER coding and six minutes on Wiki-bio coding by applying crowdsourcing strategies, e.g. we could make recommendations to our coders and simply ask them verify them; in that way they would just need to choose yes or no instead of entering detailed information. Prodigy is a promising framework that can provide us that functionality [13]. We are currently developing a more robust Arabic specific NER model to be used on our "NER and Frequency" based approaches. If we are successful, the yield and performance of the "NER" based approached will be significantly improved.

## Acknowledgment

We gratefully acknowledge the support of the National Science Foundation under award number SBE-SMA-1539302, "Modernizing Political Event Data".

## References

[1] D. J. Gerner, P. A. Schrodt, O. Yilmaz, and R. Abu-Jabr, "Conflict and mediation event observations (CAMEO): A new event data framework for the analysis of foreign policy interactions," *International Studies Association, New Orleans*, 2002.
[2] Y. Liang, A. Halterman, J. Irvine, M. Landis, P. Jalla, C. Grant, and M. Solaimani, "Adaptive scalable pipelines for political event data generation." in *Big Data (Big Data), 2017 IEEE International Conference*. IEEE, 2017, pp. 2879–2883.
[3] A. Halterman, J. Irvine, C. Grant, K. T. Jabr, and Y. Liang, "Creating an automated event data system for arabic text." in *ISA Annual Meeting 2018 in San Francisco*, 2018.
[4] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.
[5] Conflict and Mediation Event Observations, "Conflict and mediation event observations — Wikipedia, the free encyclopedia," 2018, [Online; accessed 19-June-2018]. [Online]. Available: https://en.wikipedia.org/wiki/Conflict_and_Mediation_Event_Observations
[6] M. Tomas, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representaions of words and phrases and their compositionality." in *In advances in Neural Information Processing Systems.*, 2013, pp. 3111–3119.
[7] O. Javier and A. Reyes, "Supervised event coding from text written in spanish: Introducing eventus id." in *Social Science Computer Review 35 (3)*, 2016, pp. 406–416.
[8] M. Atkinson, J. Piskorski, H. Tanev, and V. Zavarella, "On the creation of a security-related event corpus," in *Events and Stories in the News Workshop Vancouver, Canada*, 2017, pp. 59–65.
[9] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of machine learning research*, vol. 2, no. Dec, pp. 125–137, 2001.
[10] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, pp. 993–1022, 2003.
[11] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[12] M. Honnibal and I. Montani, "Spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, 2017. [Online]. Available: https://spacy.io/
[13] ——, "Prodigy: A new tool for radically efficient machine teaching," online, August 2017. [Online]. Available: https://explosion.ai/blog/prodigy-annotation-tool-active-learning