Tracking Traditions: Identifying Isnads in the OpenITI Corpus

ryan\_muther

Due to its size and coverage, the OpenITI corpus is useful for a wide variety of research purposes. Particularly, it represents an excellent opportunity to apply computational methods from natural language processing and machine learning to solve problems not only of interest to historians, but also to computer scientists. This post will focus on one problem in particular—that of detecting embedded genres in texts in the OpenITI corpus. Embedded genres are portions of a larger work that can be thought of as belonging to a genre distinct from that of the work itself, such as poetry embedded in a historical narrative or chains of transmitters (*isnad*s) embedded in a discussion of grammar. This phenomena is found throughout the corpus.

As a case study, we will focus on detecting *isnad*s: we will give an overview of machine learning and of the process used to create the training data and models for automatic identification of *isnād*s in the corpus. Unlike prior work on automatic *ḥadith* tagging, we focus exclusively on trying to identify *isnad*s, rather than also trying to simultaneously extract the corresponding stories (*matn*s) for each chain (Harrag et al, 2014; Maraoui et al, 2019, Altammami et al, 2019) or extract information from the identified *isnad*s (Siddiqui et al, 2014), as others have done. Additionally, we are working with a much larger and correspondingly more diverse collection of texts. Furthermore, much of the existing work, like that cited above, focuses on using rule-based systems to identify *isnad*s and the individual transmitters within them which limits the generalizability of these models to previously unseen texts.

The task of identifying *isnād*s in the corpus has interesting future applications, both historiographically and technically. For historians, even something as simple as the ability to identify where *isnad*s occur allows one to, for example, track how the importance of certain texts change over time based on what gets cited and how, or to understand the relative importance of *ḥadīth* citations in different texts.

Computationally, our research on *isnad*s has myriad applications to other problems in natural language processing. It lays the groundwork for improving the models used to track text reuse in the corpus, as *isnād*s currently indicate a great deal of similarity of reuse that is statistically meaningless. Additionally, it allows one to look at the problem of converting the text of an *isnad* into the network of relationships among mentioned transmitters using the *isnād*s found with this model as training data. Then, rather than having, say, a piece of text which describes that Ibn Sa’d heard some information from Ibn Ishaq, the computer would know that there are two people, one called Ibn Sa’d, the other called Ibn Ishaq and that the former received information from the latter. This would, in turn, allow scholars to apply methods from social network analysis to understand how texts were compiled and transmitted by exploring how different people were involved in the dissemination of knowledge. Similarly, this also creates a wealth of training data for named-entity tagging in classical Arabic texts, as *isnad*s are innately name-rich. This will allow later research to leverage the output of this model as a way of improving the performance of models designed to locate the names of people in a text using the names present in *isnad*s as additional training data.

Having discussed why this task is useful to resolve, we will turn our attention to how exactly we can go about performing it on a collection of texts on the scale of our corpus, with about 4,200 unique texts totaling over 1.5 billion words. In order to underscore the iterative nature of this sort of work, we will present two different approaches to the task of *isnad* detection. Both approaches involve building mathematical models of what *isnad*s look like using a set of examples created by human annotators. These models can then be used to find other *isnād*s in texts which the model has never seen before. This is an example of a “supervised problem,” where the model learns from training data, which consists of examples of what it is trying to find—in this case *isnad*s—and examples of what not to look for. (For an example of an unsupervised problem, such as test reuse detection, see the discussion in previous [blog](http://kitab-project.org/2018/05/02/detecting-what-authors-took-from-earlier-works/) [posts](http://kitab-project.org/2019/11/14/judging-the-difference-between-different-arabic-text-versions-mathematically/).) The exact form of the training data can vary depending on how the models work.

The first approach relies on the idea that *isnad*s display a characteristic distribution of particular words, including “transmissive terms” like *haddathana* or *qala*, or common name elements like “*ibn*” that can be used to identify what parts of a text are likely to be *isnad*s. Since in other contexts these terms will have a different distribution, looking for regions of text which use similar language to known *isnad*s will be enough to reliably identify them. This is done by building two language models, which are trained on different kinds of text: one is trained on examples of *isnad*s and another is trained on examples of text which are not *isnads* (the latter is called the “background”). Examples of the training data for the *isnad* model can be seen in Figure 1 below. It consists solely of a collection of *isnad*s provided by human annotators. The background model is trained on the entire corpus.

Image

Image

**Figure 1**: Training data for the *isnad* model

These models can be used to quantify the likelihood that some piece of text is an *isnad* or is not an *isnad*, respectively. Text that looks more similar to the training data used for a model will receive a higher likelihood under that model. Using these two models, we can figure out the best way to divide a new text into sections, some of which are *isnād*s and some of which are not, effectively breaking the text into a series of labeled sections of words. For example, if a certain portion of the text looks similar to the *isnad*s which were used as training data for the model, it will be labeled as an *isnad*, since it is more likely under the *isnad* model. In contrast, portions of the text that look similar to the background will be labeled as not an *isnad*, since they are more likely under the background language model.

As sensible as this approach sounds, it actually performs quite poorly since it is not very sensitive to the different contexts in which words can appear and how that affects their meaning. For instance, *qala* (“he said”) can be used both as a transmissive term within an *isnad* and as a way of introducing quotations. This model isn’t capable of distinguishing between those two situations easily.

To rectify these issues, we decided that, rather than trying to tag entire spans of words as *isnad* or not, we would instead tag individual words as part of an *isnad* or not, based on other words that surround them within some fixed window. Using this alternative method, we would be able to give the model a more nuanced understanding of context and its effect of what the correct labels for the words should be. This process, unfortunately, involves much more labor-intensive manual annotation of texts to insert *isnad* markers, as in Figure 2 below. @Isnad\_Beg@ marks the beginning of an *isnad*, while @Isnad\_End@ marks its end.

Image

Image

**Figure 2**: Manually annotated text with *isnad* markers used to train the word-tagging model — @Isnad\_Beg@ marks the beginning of an *isnad*, while @Isnad\_End@ concludes it

This text is then converted into a list of words and the correct tags associated with them, as in Figure 3. Words at the beginning of *isnad*s are tagged as “B\_Isnad”, while those that are inside an *isnād* are tagged as “I\_Isnad”. Finally, although this is not visible in the example tags below, words that are not part of an *isnad* receive the tag “O” (“Outside”). This data is then used to train the model by associating some information with each word—both about the word itself and about the frequencies of words used before and after it, within some fixed window. Computer scientists call this process “featurization”. For example, an early passage from Ibn Qutaybah’s *Gharib al-ḥadith* (below), it is tagged and featurized as in Figure 3.

قال أخبرنا أبو محمد عبد الله بن مسلم بن قتيبة

Image

Image

**Figure 3**: An example of tagged text and featurization with a window size of five words. Note how “بن” occurs twice after the current word within the window, and that is reflected in the value for the “a\_بن” feature.

During training, the model will then use this representation of the text to learn to interpret things like the presence of a high number of transmissive terms or names in the text around a word as an indicator that it is part of an *isnad*. Unlike the first approach, this model also has the ability to take context into account when the surrounding text doesn’t look like an *isnad*, and label the text accordingly. This will reduce false positive matches where, as in the example above, *qala* could be used to introduce quotations of written text. Once trained, the model can then be used to label *isnad*s in new texts by inferring the most likely tags for that new text. To do this, the text is featurized as described above and the trained model is used to find the most likely tag sequence (labeling of words as part of an *isnad* or not) for that featurized data.

To evaluate the model, we performed a process called ten-fold cross-validation: first, the data is divided into ten pieces, called folds; second, a model is trained on nine of the folds; third, the trained model is tested on the remaining tenth fold; then the process is repeated nine more times until each fold has been used as test data. This results in ten trained models and ten sets of scores, which are then averaged—as shown in the table below in Table 1.

|———–|——–|——| | Precision | Recall | F1 | | .853 | .805 | .819 |

**Table 1**: Evaluation Statistics for the Word Tagging Model

Precision and recall are two similar metrics used to measure the performance of machine learning algorithms, with a critical difference. Precision measures the percentage of words that the model tagged as part of an *isnad* that actually are part of *isnad*s. Recall, in contrast, measures the percentage of words that should be labeled as part of an *isnad* that the model labels as such.

To give a concrete explanation for what these measures mean, a model with high precision but low recall would tend to label few words as part of an *isnad*, but those that it did would usually be correctly labeled. Conversely, a high-recall, low-precision model would label many words as part of an *isnad*, finding most of the words it ought to, but also erroneously labeling many words as part of an *isnad* that are not. F1 is used to summarize precision and recall in a single value. We cannot, unfortunately, say how this compares to human results on the *isnad* detection task, as we don’t have any data on how difficult the task is for humans.

As a further experiment, we tagged the entirety of Ibn ʿAsakir (d. 571/1176)’s *Taʾrikh Madinat Dimashq* (800,000 words in total) and found that, according to our model, it contains just under ninety-one thousand *isnad*s, comprising about 40% of the text.

This work would have been impossible without close collaboration between myself and other members of the Kitab team. In particular, Professor Sarah Savant, Professor Kevin Jaques, research assistant Mathew Barber and PhD student Hassan Ahmed provided vital training data for the models discussed above.

### **Works Cited:**

Altammami, Shatha, Atwell, Eric, and Alsalka, Ammar. 2019. Text Segmentation Using N-grams to Annotate Hadith Corpus. In *Proceedings of the 3rd Workshop on Arabic Corpus Linguistics*, 31-39. ACL.

Harrag, Fouzi. 2014. Text mining approach for knowledge extraction in Sahˆıh al-Bukhari. *Computers in Human Behavior*, 30:558–566.

Maraoui, Hajer, Haddar, Kais, and Romary, Laurent. 2018. Segmentation tool for hadith corpus to generate TEI encoding. In *International Conference on Advanced Intelligent Systems and Informatics*, 252–260. Springer.

Siddiqui, Muazzam, Saleh, Mostafa, and Bagais, Ahmed. 2014. Extraction and visualization of the chain of narrators from hadiths using named entity recognition and classification. *Int. J. Comput. Linguist. Res,* 5(1):14–25.