Neural Genealogical Relation Tagging

Patrick Schone and Wesley Ackerman
{patrickjohn.schone,wesley.ackerman}@familysearch.org
FamilySearch, 50 E North Temple, Salt Lake City, UT

ABSTRACT

We have created a neural network-based relation extraction system which is able to work simultaneously in many languages without the need for multiple models. This capability extends recent advances in neural network transformers and the actual methodology does not appear to be in the literature. This neural relation tagger has greatly advanced FamilySearch's ability to mine unstructured multilingual content and extract genealogical value that can be presented to patrons. We describe the architecture of this system and then illustrate its performance in the lab. Then, and more importantly, we show how replacing our existing relation-finding component with this transformer-based one yields a substantial 54% relative performance gain.

1. BACKGROUND

For decades, volunteers and contractors have been engaged in the practice of identifying facts and associations from genealogical documents so as to make historical records findable by individuals seeking their relatives. This distillation process is typically referred to as "indexing." Over the past decade, indexing has moved from a shallow and exclusively human-labeling effort to one which is increasingly done to a faster and richer extent by machines.

One process for automatically mining content from historical data is through entity and relation extraction [1]. Entity extraction identifies key elements of interest such as persons, places, dates, occupations, etc. Relation extraction seeks to discover any connections that should be drawn between pairs of entities. For example, in relation extraction, one could have an event and a date and declare that the particular date is when the event started; or there may be two persons and the system could report that the second person is a father to the first person.

FamilySearch first started doing automatic indexing in 2015 as it applied entity and relation extraction to born-digital obituaries [2]. Since then, FamilySearch has branched into auto-indexing collections that first require automatic transcription. That is, it has published automatic indexes distilled from hundreds of millions of printed and handwritten documents up to six centuries old in many languages including English, Spanish, and Portuguese [3].

Even so, the publication of these huge volumes of documents has come with significant struggle do in one part from trying to use older, maximum entropy-based [4] (MaxEnt) relation extraction technology which we had designed originally for use with the modern English obituaries that have no recognition errors nor parsing issues. In the presence of errorful transcription in a variety of genres and an ever-increasing number of languages, the need for a more comprehensive and reliable relation tagger was clear. Additionally, with the advent of transformers [5], it seemed there should be a way to build a relation tagger that could handle all languages simultaneously and with higher overall accuracy.

We have been able to build an initial version of such a system, and this paper describes its creation and its application. We will also show how the system performs in laboratory environments using concrete examples. Lastly, and with greater importance, we show its impact on one of our major operational tasks – auto-indexing of Portuguese civil birth records – and how it in isolation yields a 54% relative performance gain over using our previously-used (MaxEnt) relation-finding component.

2. ARCHITECTURE

2.1 Data Preparation

Our neural relation system accepts as input five types of information: the tokens themselves, the characters in those tokens, any entity tags for those tokens, each entity's zero-up count in the given line of input (#0, #1, etc.) and, for people, we also include name parsing information. We sub-tokenize the input tokens using a process similar to the Bert Tokenizer [6], and we align the other four types of tags with their respective sub-tokens. Embeddings are then created for each of these five sources of input and concatenated into one master feature vector per sub-token.

The target output for each line of input is a sequence of indexes and relation tags that would be desired from the input text. For example, suppose the input text were

#0 *Bob* Jones was #1 born to #2 *John* Jones and to #3 his #4 second wife , #5 *Mary* in #6 Provo , #7 Utah on #8 27 Feb 2023.

where colors indicate different entity classes, underline indicates SURNAME, italics indicates a GIVEN_NAME, and *#n* indicates the start of the *n*th entity in the input sequence. In this situation, we want eleven relations to emerge – that Bob Jones (#0) is the principal individual in the article, that Bob has the birth_event of 'born' (#1), has a father 'John' (#2) and has a mother 'Mary' (#5); that the 'born' event happened in 'Provo' (#6) on the '27 Feb 2023' (#8); that John (#2) is the referent of 'his' (#3) and that John (#2) and Mary (#5) are spouses; that 'his' (#3) has a familymember association with 'second wife' (#4); that Mary (#5) is a member of the class of "second wife" (#4), and that 'Provo' (#6) is a subplace of 'Utah' (#7).

We represent each of these relations in one single string by providing the relative offset for the starting entity (#0, for example), the ending one (eg., #1), and the relation between them (eg., 'has event'). This string of relations is sorted by starting offsets followed by ending offsets and we require that starting offsets are always less than or equal to the ending offsets. For any relations where the end offset should actually be before the start we replace such with a reverse-form of that relationship (such as RevIsMember) so that we can ensure that ends never precede starts. Also, for equality-type relations, we augment the relations with the entity tag types that start and end the relation (meaning something like IsSamePerCor).

With these requirements, we can say that the desired relationship output sequence for the previously-given text should be:

#0 #0 IsPrincipal #0 #1 HasEvent #0 #2 HasFather

#0 #5 HasMother #1 #6 StartPlace #1 #7 StartDate

#2 #3 IsSamePerCor #2 #5 HasSpouse #3 #4 HasFamMbr

#4 #5 RevIsMember #6 #7 IsSubplace.

The input and output sequences are also wrapped with a [START] at the beginning and [END] at the end of the particular sequence. "<empty>" is used as the target when there are no desired relations.

Most transformers decode outputs one token at a time. If successful, a transformer applied to the above text sequence would be given an initial output sequence with only a "[START]" tag and would autoregressively predict that #0 should follow it, followed next by another #0, and then by IsPrincipal, and so forth until it reaches the "[END]" token. When the system trains, we do not want it to get high 'credit' for only predicting a bunch of #*n*'s, so we alter the loss and accuracy functions to account for entire triples (two indexes plus corresponding relation) before counting anything as correct.

2.2 System Design

We leverage elements of a three-layer neural transformer (seen in Figure 1) to learn to predict relations given the inputs. For our data, fewer or more than three transformer layers yielded lower accuracies. We also say "leverage" here because although we do exploit these encoders and decoders as shown in the figure, we have a number of differences that are worth commenting on.

2.2.1. Modified embeddings. The typical embedding layer of a transformer, as shown in tan color in Figure 1, will convert just tokens into a high-dimensional space. We have elements, though, that go beyond tokens alone. We have tokens, character streams, entity classes, name chunks, and the #*n* relative entity offsets. We could treat each of these as separate kinds of units and just do general embeddings, but instead we create composite embeddings that allow specific access to each of these five elements. For example, as one possible instantiation, we may have a total embedding size of 132 where 72 dimensions are token embeddings, 24 dimensions are derived from LSTMs of character embeddings, 16 dimensions are entity embeddings, 4 are name chunk embeddings, and 16 are entity offset embeddings. The size of '132' is fairly small, but this number actually works fairly well for our dataset. Yet this in only one possible setup and other sizes may work as well or better with different quantities of data. It is also feasible to just use same-sized embeddings for each of the features and add them together, but that is future study.

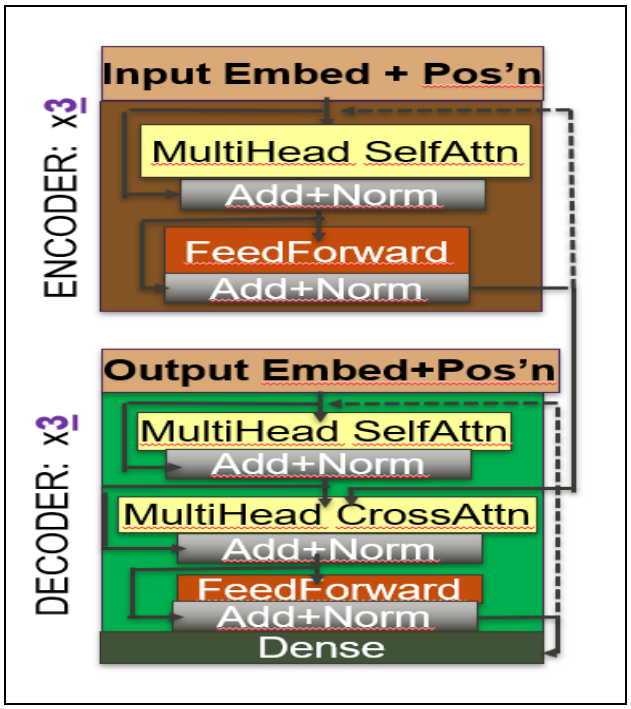

Figure 1: Key Elements of a 3-Layer General Transformer

2.2.2. Single encoder with multiple decoders. We found that if we have a single transformer try to predict all the relations, especially in a fairly data-starved environment, many desired relations will be lost. Therefore, one mechanism we used to combat this problem is to have a single encoder but multiple decoders for all non-equality relations. Particularly, we use three decoders. The first of these focuses attention on **personal and familial relations** such as the IsPrincipal, HasFather, HasMother, HasFamMbr, and other similar relations. The second focuses on relations that have to with **events**, such as HasEvent, StartPlace, StartDate. The last of the three is kind of a '**catch-all drawer**' for the relationships that do not fit into the other two categories – such as IsSubplace. This triple decoder makes fewer errors than a single decoder, and it is also slightly faster since each decoder's output sequence is shorter. On the other hand, this methodology means that the family-member relation decoder is not aware of what the event relation decoder is doing, which may impact some results.

2.2.3. Skip sequences for equalities. Transformers work with contiguous sequences of text under normal circumstances. Such was the case with the elements described in Section 2.2.2. However, if one separately accounts for coreference/equality relations in a text (like

"his" = "Bob Jones"), then the vast majority of non-equality relations can be discovered in relatively small text sequences. On the other hand, it can be the case that coreferential words can be separated by even thousands of words. However, if, say "Bob Jones" at word 20000 is the same as "Robert Jones" at word 1000, we probably do not need a lot of the intervening information to make that assessment. Instead, we just need to make sure that these two words are in some sort of text sequence together. We enable this capability by using "<SKIPx>" tokens in the text where "SKIP" indicates to the transformer that it does not get to see the intervening text and x is a integer-based logarithmic distance to show how many tokens would have been in that intervening sequence.

Therefore, for equality-style relations, we use yet another three-level transformer – with a separate encoder and single decoder – where the input sequences will include skips. We usually include several words on either side of the skip for this process. So the input to the system may be:

"Robert Jones, aged 37, <SKIP4> Bob's wife of 10 years, Harriet Jones …"

and the transformer will be asked if there are any equality-style relations in the sequence. We do not allow SKIPs to go across documents, so in this case, SKIP4 would suggest that "Bob" and Robert are fairly close-by strings within the same document. Our hope, in this case, is that it will determine that Robert and Bob are likely the same. At longer distances, this equality may not be as evident – but it may also not be as relevant, either, for extracting the kinds of genealogical content we are interested in.

2.3 Training Issues and Novelty

We train our relation tagger using 3.7M hand-labeled relations in about 40 languages (though about 15 of these languages dominate the collection). The system setups vary but typically can be fully trained in 12-48 hours on a GPU. The system learns its own weights currently though we are exploring mechanisms for getting embeddings from a much broader collection of texts and languages or from large language models available on the web. "Embedding" in this case does not mean "encoding" –it means the initial input vectors to the system.

Before leaving the subject of architecture, it is worth commenting on the novelty of this system. There have been other transformer-based relation taggers (ex [7]). Typically, these systems have the transformer predict what relation or relations might exist between a given pair of entities. However, we know of *no* relation-tagging system that will simultaneously identify all the relations for a given sequence as does ours (though [8] does simultaneous slot-filling with transformers and some of our processes are derived from that work). One-pass processing of relations as described in our paper is highly desirable both for accuracy and for speed, especially with a system like ours which has roughly 90 entity types and 25 major relation types where doing all pairs of entities would be unwieldy and even identifying one relation at a time would still be expensive.

3. PERFORMANCE: IN THE LAB

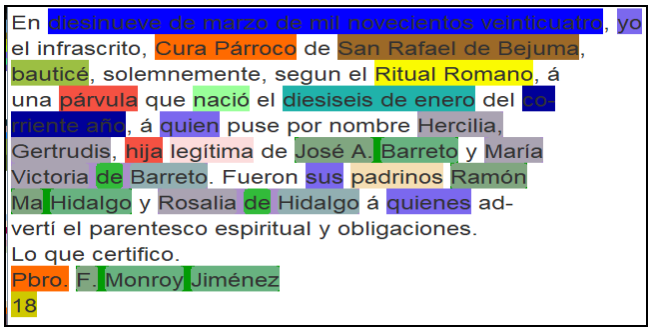

Figure 2. Entity tagged Spanish christening record

Transformers of this kind are ***trained*** by giving them the input sequence and the full output sequence up to the point of prediction. During actual ***inference***, though, all outputs must be predicted one at a time as was mentioned previously. Thus, when we train, we achieve next-token prediction triple accuracies above 93%. Yet when the full sequence must be predicted, the average triple accuracies decrease into the 70s. This may seem low, but many relations are redundant and the primary need is to at least find one instance of the relation correctly. Moreover, many of the falsely-predicted relations are not "legal" in that the entities that are paired should not produce the proposed relations, so such relations can be filtered out. (This filtering will be addressed later in this paper.) Additionally, we tag using overlapped texts in most cases, so the overlapped regions can help to identify relations that may have gotten missed in one region but not in the other.

To illustrate performance, we use the entity-tagged text of a Spanish christening record as seen in Figure 2 and apply the neural relation tagger (NRT) to that text. The predicted relations are shown below in Table 1. Relations that are tagged in green are correct. Those in blue are redundant and could be omitted. Purple indicates relations that were missed.

As can be seen from Table 1, the results looks very promising in this case – where almost all of the relations are properly produced. In fact, we see that there are no false alarms in this case, but there are the three missed relations. There are also five relations which are redundant which do not hurt performance but also do not add benefit. These redundancies came because the system has the separate modules for equality and non-equality, so it cannot guarantee that these relationships are already there – and thus it reports them twice.

From Table 1, we also see that the offsets are absolute numbers here rather than zero-up relative offsets we have been using up to this point. When we produce our input strings to the transformer, we provide it with the relative offsets and we store separately the relative-to-absolute offsets. So after the transformer predicts something like #1 and #2, we can convert these to their true offsets such as 56 and 75.

Relation	StartWord	Offset	End Word	Offset
HAS FACT	yo	56	Cura Párroco	75
RESIDENCE PLACE	yo	56	San Rafael de Bejuma	91
HAS FACT	yo	56	Pbro.	471
HAS TITLE	yo	56	Pbro.	471
STARTDATE	bauticé	113	diesinueve de marzo … veinticuatro	3
STARTPLACE	bauticé	113	San Rafael de Bejuma	91
HAS FACT	bauticé	113	18	495
STARTDATE	nació	178	diesiseis de enero	187
SUBDATE OF	diesiseis de enero	187	co-\nrriente año	210
HAS EVENT	quien	229	bauticé	113
MEMBER OF	quien	229	párvula	166
HAS EVENT	quien	229	nació	178
SAME AS	Hercilia,\n Gertrudis	251	quien	229
PRINCIPAL PERSON	Hercilia,\n Gertrudis	251	Hercilia, Gertrudis	251
MEMBER OF	Hercilia,\n Gertrudis	251	hija	272
HAS FATHER	Hercilia,\n Gertrudis	251	José A. Barreto	289
HAS MOTHER	Hercilia,\n Gertrudis	251	María\nVictoria de Barreto	307
HAS FACT	hija	272	legítima	277
HAS_FAMLY	José A. Barreto	289	párvula	166
HAS_FAMLY	José A. Barreto	289	hija	272
SPOUSE	José A. Barreto	289	María\nVictoria de Barreto	307
SAME AS	sus	341	quien	229
HAS_FAMLY	sus	341	padrinos	345
MEMBEROF	Ramon\nMa Hidalgo	354	padrinos	345
MEMBEROF	Rosalia de Hidalgo	373	padrinos	345
MEMBEROF	Ramon\nMa Hidalgo	354	quienes	394
MEMBEROF	Rosalia de Hidalgo	373	quienes	394
SAME_AS	F.Monroy Jiménez	477	yo	56
5xREDUND'T	sus/Hercilia's father, mother; yo/F.Monroy's elemts			
3xMISSING	Familymbr(Rosalia,párvula/hija), Fact(Hercilia,18)			

Table 1. Neural relation predictions on text from Figure 2

4. PERFORMANCE: THE REAL TASK

Although the lab results look beneficial, the true test is how well the system performs on a real problem. Specifically, how good do the predicted relations look when generated for actual genealogical records? Do these models give us significant gains over MaxEnt on a large-scale dataset of fully structured records?

4.1. First Contact

It is said that no system survives first contact with real data, so we expect an iterative process will be required to optimize the use of these neural relations. Even so, Table 2 shows this 'first contact' performance of our system (NRT) compared to the previous Maximum Entropy (MaxEnt) baseline that we had formerly. In this situation, we run both systems against a test set of 943 Portuguese-language civil birth records that have been fully structured and annotated for all names, dates, relationships, and places.

Performance is measured using a weighted F-score. Each expected record element (e.g., name, date, place, event, relationship type, relationship member, occupation, etc.) is assigned a weight based on its estimated genealogical relevance. Information on the main subject of the record is weighted higher, and information about unrelated people is weighted lower. Information about close family relationships (parent-child, spouse, grandparent) and vital events (birth, marriage, death) is valued more highly than other types of relationship or event information.

These models are being tested in isolation, without any fixes or post-processing applied to their outputs. When using our older, original MaxEnt models, we obtained a great deal of our quality by leveraging rules and other post-processing steps. Yet the large improvements we show here in weighted F-score (Table 2) indicate that our neural relation tagger (NRT) models greatly improve relation quality which decreases the need for post-processing to make generated relations usable by downstream processes.

Model	F-score	Delta
Baseline (MaxEnt)	48.09	N/A
NRT (Ours)	69.17	**+21.08**

Table 2. First-Contact Replacement Attempt

As shown, there is a sizeable gain in performance using the new models over the MaxEnt models. There is an *absolute* gain of 21.08%, and this equates to a 41.45% increase in *relative* accuracy. Moreover, the MaxEnt system that was applied was trained exclusively on Spanish and Portuguese records. Attempts to widen its training set to other languages yielded poorer performance and more corrupt relations. Our NRT model, on the other hand, includes high volumes of relation-tagging data in English, Chinese, Scandinavian languages, Russian, and other languages. Not only does our model generate significantly higher quality relations, but given these additional languages of training, it

is likely to be substantially more robust and capable of operating well in these other languages without the need for additional models.

4.2. Refinement #1: Probability Thresholding

Although the initial models perform much better than MaxEnt, they are not without their problems. The most prominent problem is a tendency to hallucinate additional relations. These incorrect relations are often extra occupations or residence-place facts, or erroneous parent-child relationships that falsely connect different people in a record. These incorrect relations often come associated with lower probability scores from the transformer.

When our older MaxEnt system is run, we use various thresholds to determine whether it is confident that a given relation exists. This is not perceived of as 'post-processing' for MaxEnt but rather as determining its best usage settings. We felt that a similar tuning/filtering approach should be applied here on the NRT's output: such filtering could hopefully prune back any low-confidence relations that the transformer has proposed.

Entity-Pair Probability Filtering. As was said, the transformer predicts each output token one at a time. For each relation generated, it predicts a starting entity offset, followed by an ending entity offset, and then predicts the relation it thinks connects those two offsets. Suppose that it predicts #5 as the first offset and #7 as a second offset, but it is only 75% confident that #7 should follow #5. Perhaps this is insufficient as an entity-pair confidence and the whole predicted relationship should be thrown out. Only using entity pairs that the model is highly confident about has potential to improve relation quality.

Relation Choice Probability Filtering. Even if we suppose that the system is confident about an entity pair, the neural network may still not be very confident about the relation it has chosen to bind those two entities. In such case, we may want to throw out those relations as well.

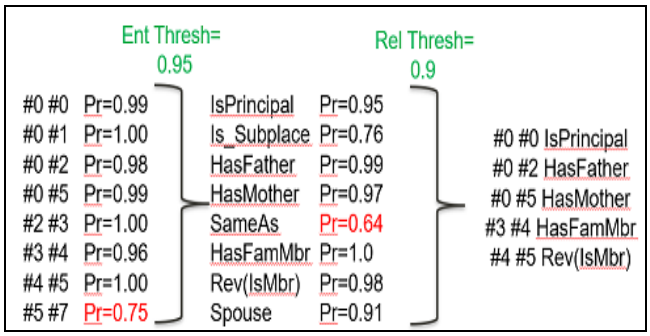

Figure 3. Entity and Relation Filtering

The illustration shown in Figure 3 illustrates how these two filters may be applied. The system may receive the entity pairs depicted in the left of the image, but if we suppose that the threshold for acceptance is 0.95, the #5 #7 entity pair will need to be rejected. Then, it may be that the #2 #3 entity pair is assumed to be highly likely by the system, but the relationship that is proposed ("SameAs" in this case) has a probability that is too low for the relation threshold. Given these two settings, the final set of accepted relations are those shown in the right side of Figure 3.

Table 3 shows the effects on weighted F-score of applying these two separate probability thresholds. Setting an entity pair threshold of 0.95 with a relation threshold of 0.9 yields an additional 4.46% points of accuracy over NRT without any tuning – bringing the total gains over MaxEnt to 25.54%.

Model	Entity Pair Thresh	Relation Thresh	F-score	Delta
MaxEnt	N/A	N/A	48.09	N/A
NRT (Ours)	-	-	69.17	+21.08
NRT (Ours)	-	0.7	69.38	+21.29
NRT (Ours)	0.9	-	73.29	+25.20
NRT (Ours)	0.7	0.9	72.96	+24.57
NRT (Ours)	0.9	0.9	73.26	+25.17
NRT (Ours)	0.95	0.9	73.63	**+25.54**

Table 3. Entity and Relation Probability Thresholding

As can be seen from Table 3, thresholding on entity pair confidence is more important by far. Using only this threshold, with a setting of 0.9, we obtained a 4.12% absolute increase in F-score. This indicates that there were many cases where the model was generating relations when it was less confident that a given entity pair were even associated at all. The entity-pair threshold drastically pruned the extra erroneous relations that the untuned models were producing. We found it was a rapid way to refine and regularize the generated relations with very little logic required.

Thresholding on relation type confidence proved only slightly useful. It contributed approximately 0.2% improvement to the F-score. We felt this was intuitive: if a model selects a "bad" entity pair, it may still be very confident in the relation that should connect them since there are relatively few relation types that make sense for two given entity types. From what we have observed so far, entity-pair confidence thresholding is required to ensure relation quality, but relation type thresholding seems to only clean up a small number of anomalous relations.

4.3. Refinement #2: Class Filtering

Another kind of filtering we had been using in the MaxEnt system was that it was only allowed to consider those relations that could "legally" exist between a given pair of entities. For example, a relationship such as "IS_SUBDATE" which can only be applicable between temporal elements should not be allowed to have starting and ending entities like PERSON and LOCALE, respectively. It seemed reasonable to apply these same kinds of constraints to the output of the NRT.

There is, however, a scenario with the NRT that was not observed much with MaxEnt models. Since we require the transformer of our system to have the first entity always precede the second (which we mentioned earlier in connection to reversed relations), it is possible that the system proposes the entities out of order. For example, if the first entity is "DATE" and the second is "EVENT" and the relation is HAS_START_DATE, we possibly can conclude that the entities got accidentally reversed. We define filtering templates to reverse entity positions when they appear to be in reversed order.

Table 4 shows the effects of adding this consistency checking/class filtering. We can see that both kinds of filtering help the system individually and that they are complementary and yield even a better result when used together. With both probability thresholding and class filtering in place, our overall performance increases another 0.6% in absolute F-score – yielding an overall absolute improvement of 26.14% above our MaxEnt models.

Model	**Remove Malformed Relations**	**Swap Reversed Relations**	**F-score**	**Delta**
MaxEnt	N/A	N/A	48.09	N/A
NRT (Ours)	No	No	73.63	+25.54
NRT (Ours)	Yes	No	74.11	+26.02
NRT (Ours)	No	Yes	73.98	+25.89
NRT (Ours)	Yes	Yes	74.23	**+26.14**

Table 4. Entity and Relation Probability Thresholding

5. COMMENTS

The final score for our transformer models is 74.23%, a 26.14% absolute improvement – and a **54% relative improvement over our MaxEnt models.** Again, this score is of each model in isolation, without downstream rules, auxiliary support processes, or complex refining logic. Our full MaxEnt system requires a great deal of handcrafted, complex logic for **each language and record type** to refine the generated relations so that they could be utilized by downstream processes. These new transformer models perform much better out-of-the-box on much larger variety of data, they save engineering time previously needed to integrate many models for different languages, and they reduce the engineering time previously needed to hand-craft logic to refine relations. While logic and rules could still further refine the output of these models, such things are now less of a necessity and more of a "if desired."

In this effort, we had begun with a hope to replace MaxEnt as a relation-tagging mechanism with a newer kind of transformer-based relation tagger. We demonstrated that this change has given us a sizeable boost in accuracy while at the same time providing a system that is able to handle languages more universally. We have made use of these upgrades to successfully process tens of millions of historical Portuguese Civil Births.

That said, the numbers we saw in Table 4, though better, still have room for improvement. We are currently developing larger transformers that have been training on significantly more text and other textual phenomena as a means of potentially getting even bigger performance boosts in the near future.

REFERENCES

[1] Grisham, R. & Sundheim, B. (1996). Message understanding conference -6: A brief history. *16th conference on Computational Linguistics* (pp. 466-471).

[2] Schone, P. & Gehring, J. (2016). Genealogical Indexing of Obituaries Using Automatic Processes. *FHTW*. https://fhtw.byu.edu/static/conf/2016/schone-indexing-fhtw2016.pdf

[3] Schone, P. (2020). Economical bimodal classification of a massive heterogeneous document collection. *FHTW*. https://fhtw.byu.edu/static/conf/2020/schone-economical-presentation-fhtw2020.pdf

[4] McKallum, A.K. (2002) MALLET: A machine learning for language toolkit. *http://www.cs.umass.edu/mallet*

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, AN, Kaiser, L., Polosukhin, I. (2017) Attention is all you need. *NIPS*.

[6] Devlin, J., Cheng, MW, Lee, K., Toutanova, K. (2019) BERT: Pre-training of deep bi-directional transformers for language understanding. *ACL* (pp. 4171-4186).

[7] Zhong, Z., Chen, D. (2021). A frustratingly easy approach for entity and relation extraction. ACL (pp. 50-61).

[8] Kerman, S. (2022). Internal communications (SWFTY, Transformer-based Slot Filling), FamilySearch.