A Multilingual Personal Name Treebank to Assist Genealogical Name Processing

Patrick Schone and Stuart Davey
FamilySearch, 50 E North Temple, Salt Lake City, UT
Patrickjohn.Schone@ldschurch.org, DaveySE@ familysearch.org

ABSTRACT

In this paper, we illustrate the creation of a completely new Treebank which has significant application to the genealogical space and, to the best of our knowledge, has never been described before. We document the creation of a *Personal Name Treebank (PNTB)* which, though still a work in progress, already contains over 150,000 name structure classifications for people names derived from all the cultures, time frames, and writing scripts that are observed in our 800-million-name Common Pedigree at new.familysearch.org. The Common Pedigree includes names from various millennia, name from all countries of the world, and names rendered not only in Latin, but also in scripts such as Cyrillic and CJK. We describe the PNTB and its components, and we give a number of examples where this is particularly beneficial to genealogical search.

1 BACKGROUND

In name processing, it is customary for a name service to throw out punctuation, tokenize personal names at white space boundaries, and identify each name component as either a “given name,” a “surname,” or some other kind of name. For many types of name processing, this strategy works reasonably well, especially for modern Anglo-centric names.

This name-handling strategy is usually also followed in genealogical name search, which is our particular interest. Unfortunately, when the search for personal names extends beyond the bounds of modern English origin, the simplifications described above begin to break down and result in search errors.

For example, in most of Scandinavia (and even still today in Iceland), people were named according to a patronymic system. Johan, the son of Erik, might accordingly be named “Johan Eriksson.” Since there could be many Johans whose fathers were each Erik, Scandinavians might distinguish this Johan Eriksson by the farm he lived on, such as “Johan Eriksson Holm.” Using the Anglo-centric simplification for name handling, one might treat “Holm” as the ‘surname.’ However, not all Scandinavian records will refer to Johan by the name of his farm – they may only represent him as “Johan Eriksson.” If his name is stored in a genealogical search engine under the surname “Holm,” he would not be found. Also, if the record indicates that his surname is ‘Eriksson,’ there is a very real chance that his father, Erik, has a surname other than “Eriksson.”

In Spanish, Portuguese, and other languages of Latin origin, a given name can be represented as multiple words such as “Maria de la Cruz”. It can be sufficient to treat this name as two key components, “Maria” and “Cruz” and a search engine would probably find the name. If a genealogical search engine takes advantage of name conflation/expansion, treating this name by its parts is less beneficial. For example, it could be that the person also went by “Maricruz.” “Maricruz” is neither a conflation of the name “Maria” nor of “Cruz” but of the entire name phrase. In addition to given name issues, these languages frequently identify multiple surnames to indicate the individual’s father’s family name, mother’s family name, and/or the spouse’s family name. It is not unreasonable for a Spanish name to be *María de la Cruz Juana Gómez Velásquez vda. de Gutiérrez.*

These interesting naming phenomena exist in many language and across time periods. Rather than coerce names to fit an Anglo-centric model, it might make more sense if a genealogical search engine could leverage a personal name parse. The parsers job would be to accurately describe all of the phrases and constructs of the name, which could greatly facilitate search.

At the very end of the 1980s, the University of Pennsylvania embarked on an effort to create full syntactic parses of English sentences [1]. The collection of these syntactic parses became known as a *Treebank*. Once the first Treebank in English was created, it became clear that these kinds of resources which could in turn be used for creating or training rule-based and statistical parsers were of great interest. Therefore, treebanks began to proliferate. Now they exist in many languages and for many different genre of texts.

To the best of our knowledge and from what we can find, however, there has never been a treebank that was created specifically for parsing personal names. Consequently, we introduce our creation of a ***Personal Name Treebank*** (***PNTB***). Our PNTB incorporates modern and historical personal names across many language boundaries. These names are drawn from the personal names in the LDS Church’s “Common Pedigree” (available at new.familysearch.org) which is quickly approach a billion names. The common pedigree contains names from many cultures of the world and some that date back for millennia, in addition to names that are in non-Latin scripts (particularly Cyrillic and CJK languages).

Though the PNTB is currently under development and the annotation project is expect to continue for the next few months, at the time of this paper, the PNTB already contains over 150,000 name structure classifications. We plan to extend existing parsers using the PNTB for application to genealogical name searching. We would also like to get the PNTB into the hands of researchers around the world. We therefore provide a description of the creation, features, and attributes of the current instance of the PNTB.

2 A BRIEF LOOK AT TREEBANKS

Wikipedia provides an appropriate definition of a treebank: "*A treebank or parsed corpus is a text corpus in which each sentence has been parsed, i.e., annotated with structure*" [2]. Depending on their elementary school, many children are often asked at an early age to do "sentence diagramming," which is effectively identifying the parse structure of a sentence. A treebank is a large collection of these diagrammed sentences. Since we will be describing the parsing of names later on, it is prudent to take a moment to describe at least some of the components and vocabulary associated with the parsing process.

2.1. An Example of Sentence Parsing

Consider the sentence:

"The brown fox jumped over the lazy dog."

The words "the" in each case of this sentence are determiners, and are typically marked as "**DT**." The words "brown," and "lazy" are adjectives. These are typically expressed in treebanks by the symbol "**JJ**." The words "fox" and "dog" are nouns, and are represented by "**NN**." "Jumped" is a past tense verb which may get marked as "**VBD**," and over is a preposition, or "**IN**". The period at the end is usually just tagged as "**.**"

If the sentence can be relabeled to include these tags:

The/DT brown/JJ fox/NN jumped/VBD over/IN the/DT lazy/JJ dog/NN ./.

When these tags are applied to the sentence, we would say that the sentence is tagged for *part of speech*.

A parse goes beyond this process and converts the sentence into its nested or linked components. Parses that show nesting are usually called "*constituency parses*" and those that show linkage are "*dependency parses*." For the moment, we will consider our example sentence by its constituency parse. The fox is brown and the dog is lazy, and we would like to bind these facts together. Usually, adjectives and determiners are brought together into noun phrases, **NP**. In this case, we have the first noun phrase [NP [DT The] [JJ brown] [NN fox]] and another [NP [DT the] [JJ lazy] [NN dog]]. The preposition "over" usually signals the existence of a prepositional phrase, **PP**, which would be represented as

[PP [IN over] [NP [DT the] [JJ lazy] [NN dog]]].

The word "jumped" signals the event of a verb phrase, **VP**, which might be given as [VP [VBP jumped] [PP …]]. Lastly, we would combine the first NP with the VP and produce a sentence, [**S** [NP …] [VP …] [. .]]. Once the phrase has been packaged up to the sentence level, the sentence is parsed according to its constituencies.

Constituency parses sometimes also indicate the *head* word for each phrase. The head is the key ingredient of the phrase. In the noun phrase [NP [DT The] [JJ brown] [NN fox]], the head is "fox" and in the verb phrase, the head is "jumped." This suggests that "fox jumped" is skeletally similar to the full sentence.

2.2. Constituencies or Dependencies

Our example shows that through a parser, one can take a larger sentence and prove that, at heart, it has a fairly simple structure. The sentence we just saw rolls up to have the same structure as the extremely basic sentence, "John reads." -- that is [S NP VP .]

An appeal of treebanks is that they are built on broad ranges of sentences (including incomplete sentences), so they can be used for helping to derive the meaning of even the most complex sentences. A number of people have developed computational systems for ingesting treebanks and creating automatic parsers (such as Charniak [3] and Collins [4]). Through the use of these treebank-derived parsers, very sophisticated machine systems can be created which can ingest 1000s of sentences and can attempt to get at the understanding of large collections of text.

A drawback of constituency parses such as those that were shown is that they can be complicated and costly, and they provide more information and detail than what is often needed for understanding. Dependency parsers are simpler. They merely link components such as "the" and "brown" to the word that they depend on, "fox." They also do not requiring any particular ordering of words. However, a constituency parse can be turned into a dependency parse but the reciprocal is not usually true. The constituency parse preserves order, which may be useful, and the phrases themselves can be important.

3 FEATURES OF THE PNTB

We now describe the current ingredients of the PNTB. We believe that phrasing and order is important. Therefore, we have created the PNTB as a constituency parsed treebank.

This section includes definitions of the constructs for the "personal name parts of speech" and genealogical parse structure. To provide better clarity of how the definitions apply to name parsing, we will illustrate the use of each of these constructs by showing examples.

A warning is in order. Section 3 is appendix-like in nature as it walks though each of the constructs for use in the PNTB. These constructs will be referred to in subsequent sections and have therefore been identified here. Yet for readers who are primarily interested in the PNTB creation process itself and of current PNTB statistics, they may choose to review Section 3.1 for a key name parsing example and then may proceed to Section 4.

3.1. Simple versus Complex Names

When linguistic treebanks are created, they almost always bypass the internal structure of personal names. Current parsers would treat names in terms of the parts of speech of the constituent name parts. For example, the name "John Smith" would usually be marked as a noun phrase (NP) involving two tokens "John" and "Smith" which are proper nouns (**NNP**). Its constituency parse would be represented as [NP [NNP John] [NNP Smith]].

The PNTB can serve as an extension to existing parsers when it comes to handling these personal names. For genealogical purposes, we prefer to parse names in ways that help to identify the genealogical syntax roles that the name pieces provide.

Let us begin here to identify relevant parse structures that would be desirable for genealogical handling of names. Let us start with some initial definitions:

GN= "Given Name" (a name that is given to the individual)

GNP = Given Name Phrase

FN="Family Name," with the following subsets

FNF = "Family Name of the Father"

FNM = "Family Name of the Mother"

FNS = "Family Name of the Spouse"

FND = "Name derived from a family name"

SNP = "Surnominal Phrase"

Using these definitions alone, we can provide the more useful genealogical parse for "John Smith" of

[NAME [GNP [GN John]] [SNP [FN Smith]]],

or, if we know more information about the individual, the parse could also be

[NAME [GNP [GN John]] [SNP [FNF Smith]]] .

An initial reaction to this parse might be "Have we handled this name better through using this parse than through using existing practices?" This concern may even be strengthened by the fact that, in genealogical data, a full personal name may actually be pre-delimited (though possibly incorrectly) by a genealogical patron as "John /Smith/" to make handling even easier.

It would certainly be hard to contend in favor of a treebank if all names were as easy as "John Smith" to parse. Indeed, it is the case that names with "genealogical parts of speech" "GN FN" *are* the most frequent – at least this is the case in our huge genealogical repository.

Yet there are thousands of different name structures that occur in the data. For example, we previously identified a viable name, *María de la Cruz Juana Gómez Velásquez vda. de Gutiérrez.* This name has more parts than we have yet identified. The word "vda.," means "viuda" or "widow" and it constitutes a bound particle that suggests that this woman is associated with a deceased spouse. Suppose we provide the following definitions:

REL = Relational particle (a word that signifies a relation with one of the other name parts), with subsets

RELW = "Wife-indicating relational particle"

RELS = "Son-indicating relational particle"

RELD = "Daughter-indicating relation particle"

RELM = "Mother-indicating relational particle"

RELF = "Father-indicating relation particle"

RELC = "Genderless child relation particle"

RELV = "Servant relation particle"

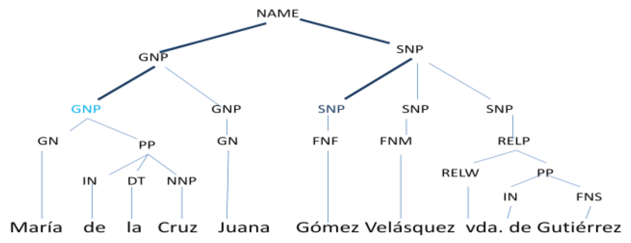
RELP = Relational Phrase

The term "vda." would have a genealogical part of speech of RELW. *Gutiérrez* is the woman's husband's name, and "de" is a preposition or "IN" (which we saw earlier), we could form a relational phrase

[RELP [RELW vda.] [PP [IN de] [FNS Gutiérrez]].

"Gómez" and "Velásquez" are family names passed to the individual, respectively, from the father and the mother.

Lastly, though "Juana" is a simple GN, "María de la Cruz" is an entire phrase. The name María is both a GN and is clearly the head of the phrase. "Cruz" is a noun-name (meaning "cross"), so it could be represented either as another given name or as a proper noun. Altogether, a parse for this name might be represented graphically by something like the following figure:



Since there are many names that might that differ in simplicity from the "John Smith" case, we will proceed to identify such cases and define parsing constructs that will allow us to incorporate such names into our PNTB.

3.2 Patronymic Names

We mentioned the example of "Johan Eriksson" before. In that case, "Eriksson" was not a family name that was passed from generation to generation. It was a *patronymic*, a name associated with the father's given name. In that sense, one could mark the "-sson" suffix of the name with RELS and "Erik" as some kind of given name. Since it is not the individual's given name but rather that of the father, we will introduce the tags

GNF = "Given name of the Father"

GNM= "Given name of the Mother"

GNS = "Given name of the Son"

GND = "Given name of the Daughter"

GNS = "Given name of the Spouse"

GNV = "Given name of the Person Served"

Consequently, "Eriksson" could be turned into a RELP,

[RELP [GNF Erik] [RELS –sson]].

However, many countries that had used patronymics began to eliminate them in the mid-1800s. So if one sees the personal name "Johan Eriksson" without having any knowledge of the familial relations, it is difficult to know whether the name is patronymic or if it has evolved into a typical surname. Although it is our plan to eventually mine the *relationships* from Common Pedigree (in addition to merely considering the names) and determine from those relationships if such names are indeed patronymic, we will tag likely-patronymic names generically as

PN = "Patronymic Name (of origin)" or

PM = "Matronymic Name (of origin)"

and include them in a

PNP = "Patronymic Phrase."

These construct will also be used when it is unclear what potion of the name is the father's or mother's name. With these constructs, we should also be able to handle names from other cultures such as "David ben Jesse" and "Pavel Aleksandrovich."

3.2.1. Patronyms but Exceptions

The reader may know that names that we saw in Section 3.1 like Gómez and Velásquez are themselves patronymic in origin. However, it has been almost 500 years since such names were regularly patronymic, so we will treat these names as indicated before unless we obtain relational verification from Common Pedigree which suggests otherwise.

"Fitz" and "Mc" are also ancient particles indicating patronymic relationships that have been out of use for at least two centuries. So we will treat these as particles, as described in Section 3.6, unless we have relational evidence of them being bound.

3.3 Honorifics and Titles

People are commonly referred to by honorifics and titles. In the names Dr. John /Smith/ and Mrs. Judy /Jones/, the words "Dr." and "Mrs." are honorifics. Honorifics are not permanent parts of the person's name, but they provide a notion of respect. We will convey honorifics by

H = Generic or male honorific

HF =Expressly female honorific.

It may seem unusual to specifically identify a female honorific when we have not included gender in other tags. However, there is value to doing this beyond preservation of gender information. Consider the three names:

Mrs. Judy Jones

Mrs. John Jones

Mrs. Pat Jones

In the first instance, the name "Judy Jones" is clearly female and the "Mrs." indicates that Judy Jones is married. In the second instance, though, there probably is not a woman John Jones, but rather, this "Mrs." says that there is a woman whose name is unknown and she is married to John Jones. In the third case, "Pat" is a gender-neutral name and it is not clear if "Pat Jones" is the woman or the husband. By using "HF" as a tag, we can hedge bets.

There are other special honorifics that are actually associated with a position of rule. Examples of this might be "Queen," "Prince," and so forth. We will tag these with

T = Title

and we will allow honorifics or titles to be included in titular phrases, that

TP = Titular Phrase.

Titular phrases can also include multiword phrases such as "Her Royal Highness" or "His Majesty."

3.4. Ordinals and Generationals

As we mention royalty, it is also advantageous to talk about ordinals. Consider the name "King George III." The "III" indicates that there were probably two other kings named George who existed before the specified one. We will create an ordinal class,

ORD, with subclasses

ORDR: An ordinal expressed by Roman Numerals

ORDG: Ordinal expressing generational differences

The string "III" associated with King George would be tagged as ORDR.

Note, though, that there could be a person name "George Smith III" -- the potential son of "George Smith Jr." Though "III" will be tagged as ORDR, we will tag "Jr." as an example of ORDG.

3.5 Locative/Toponymic Names

King George III may actually have been observed as "King George III of England." "England" is a toponymic name – one that refers to a location associated with the individual. We will refer to "England" by

NNPL= Location

We will treat "of England" as a special kind of prepositional phrase, and will refer to it as

PPTP = Toponymic Prepositional Phrase.

Identifying the places associated with an individual is difficult, and, as we have said before, we will eventually need to appeal to the Common Pedigree relationships and auxiliary information in order to help determine when a phrase is toponymic versus when it is merely a surname.

For example, the phrase "Van Roosendaal," which translates to "Of Rose Valley," is a locative phrase. Dutch, German, and various other languages use these kinds of name constructs. However, just like patronymics which *may* have lost their original origins, names like "Van *X*" may be perceived of as merely a surname. Therefore, unless we know that the location being identified in a name is making specific reference to a place associated with the person, we will assume a name like “Van Roosendaal” is a multi-part surname with a bounded component (see 3.6).

3.6 Bounded Name Components

It was mentioned originally that many name processing algorithms break names at whitespace boundaries. This can cause significant problems when genealogically searching for people with multi-part names.

In Section 3.2, we talked about name pieces such as “Mc” and “Fitz” which are particles that cannot stand on their own but which are bound to another word. For example, if a genealogical patron is interested in the name “Mc Coy,” failing to associate “Mc” with “Coy” will devolve the query into a search for “Coy.” Though this may still yield reasonable results, a query for “John Fitz William,” will probably be less successful because either the term “Fitz” will be eliminated from the query altogether or it will be treated as a search term and allow every “John Fitz Something” to become a potential response.

Additionally, in Section 3.5, we talked about “Van Roosendaal” and other names like “Van X” and “Von Y.” These names have similar issues to “Mc Coy” and “Fitz William,” as do name phrases like the Portuguese “Da Costa” and the Spanish “De La Rosa.”

Another example of interest comes from a name like “Jill St. John.” “St.,” which is an abbreviation for “Saint,” should not be part of a name search without being attached to “John.” It would be unacceptable for a query to return either “Jill. St. Mark” or “Jill M. John.”

We will call a word piece *bounded* if it should not exist without the name component to which it is attached. In the cases of “Van Roosendaal” and “Da Costa,” we can bound the pieces if we treated them initially as prepositional phrases which are then wrapped in a larger phrase, such as

[FNP [PP [IN Da] [NNP Costa]] and

[FNP [PP [IN Van] [NNP Roosendaal]]].

We define **FNP** = Family Name Phrase, with subtypes of

FNPF = Father’s Family Name Phrase

FNPM = Mother’s Family Name Phrase

FNPS = Spouse’s Family Name Phrase

These constructs accommodate the handling of some bounded conditions. They do not allow us to tackle “Mc,” “Fitz,” or “St.”. So we introduce two additional particles,

PARTRB = Right-bound name particle

PARTLB = Left-bound name particle

These allow us to bind a particle to a token to the right or to the left. Thus, we would represent the surname “Fitz William” as [FNP [PARTRB Fitz] [NNP William]]. For consistency, if the name had appeared as “FitzWilliam” or “Fitzwilliam”, we will represent these respectively as

[FNP [PARTRB Fitz_] [NNP William]], and

[FNP [PARTRB Fitz_] [NNP william]].

3.7 Alternations and Conjuncts

Earlier, we saw an example of a woman whose parents names were Gómez and Velásquez. In some languages, these names would not appear juxtaposed, but rather, they would be combined with a conjunction such as

“Gómez Y Velásquez”

where “Y” means “and.” Conjunctions also appear in surnames such as “Natt och Dag” (“Night and Day”).

Conjunctions (and disjunctions) also show up in a number of other genealogical names when the exact spellings of names are unknown, or when names in non-Latin scripted are transliterated. Three examples are “Jack or John /Smith/,” “Jean (Jane) Purdie,” and “郭KWOK.”

In normal treebanks, “CC” already is a part of speech to indicate a conjunction. We will use CC specifically for “and” conditions, and we will use **CCD** for disjunctions (“or” conditions). CC (or CCD) are usually used to conjoin two phrases of the same type.

The parentheticals already indicate disjunction. Often parentheses are represented as “**-LRB-**” and “**-RRB-**” (left and right round brackets) to avoid confusion with the parsing brackets. The parentheses (or LRB/RRB) are their own parts of speech.

In the case of the transliteration, we will introduce the **EQUALS** construct for showing equality assignment between words.

Using these constructs, we can parse our five examples as:

[SNP [SNP [FNF Gómez]] [CC Y] [SNP [FNF Velásquez]],

[SNP [FNPF [NNP Natt] [CC och] [NNP Dag]]],

[GNP [GN Jack] [CCD or] [GN John]] [SNP [FN Smith]],

[GNP [GN Jean] [((] [GN Jane] [))]] [SNP [FN Purdie]],

and [SNP [FNP [EQUALS [FN 郭] [FN KWOK]]].

3.8. Wildcards and Arbitrary Fields

Up to this point, we have been identifying relatively common personal name constructs. There are a number of additional name structures that occur frequently in genealogical databases but which are not actual full names. These fall into four categories:

[1] Initialisms, such as the “F.” of “John F. /Kennedy/;”

[2] Abbreviations, such as the “Ma.” in “Ma. /Lopes/;”

[3] Wild cards, such as the “…” in “Fr…d G. /Williams/;”

[4] “Arbitrary” names, such as “Baby” or “Anonymous” in the phrases “Baby /Jones/” and “Anonymous /Smith/.”

We will give each of these a separate personal name part of speech tag. The first several are kinds of abbreviation, **ABBR**, and we subdivide them into

ABBRI = Abbreviation as an initial,

ABBRGN = An abbreviated given name,

ABBRFN = An abbreviated family name, and

ABBRWC = Abbreviation containing wild cards.

The fourth category can either fall into a relational class (as seen in Section 3.1) or it behaves as a class unto itself. To handle it, we introduce the constructs

X = Place holder for an unknown name, and

XP = Place holder phrase.

The five examples can be parsed as follows:

[NAME [GNP [GN John] [ABBRI F.]] [SNP [FNF Kennedy]]]],

[NAME [GNP [ABBRGN Ma.]] [SNP [FN Lopes]]],

[NAME [GNP[ABBRWC Fr*d] [ABBRI F.]] [SNP [FNF Williams]],

[NAME [RELP [RELC Baby]] [SNP [FN Jones]]], and

[NAME [XP [X Anonymous]] [SNP [FN Jones]]].

3.9 Phrasal and Syllabic Names

As name processing extends to names from Asian cultures, aboriginal languages, Native American tribes, and so forth, it can be the case that an entire of string of names must be treated as a single unit. For examples, the following are names that contain multiword parts which must be treated as whole units: "CHAE GI-YEONG," "/甯/ 小 姐," "Crow Flies High," and "Barbara Mesh ke ah ko quah."

If a genealogical patron is interested in finding his relative "Crow Flies High," is would be unacceptable to return another name containing a subset of this phrase. Perhaps the only acceptable name variant that might be appropriate would be to return a Native American word which translates into "Crow Flies High." If we use the usual parse construct **NNP** to mean a proper noun, we can embed it into a GNP to provide a parse for Crow Flies High:

[NAME [GNP [NNP Crow] [NNP Flies] [NNP High]]].

We could conceivably repeat this process for each of the phrasal names. But there is a distinction. "Crow Flies High" is a semantic rendering of a Native American name, and its pieces are proper nouns. However, the partial name "Mesh ke ah ko quah" is a sequence of syllables which have no meaning in English and may have no particular meaning in isolation in their language of origin. Since this part of the name is composed of a sequence of syllables, we will represent each piece by the construct **SYL**. Therefore, we can parse the name "Barbara Mesh ke ah ko quah" as,

[NAME [GNP

[GNP [GN Barbara]]

[GNP [SYL Mesh] [SYL ke] [SYL ah] [SYL ko] [SYL quah]]].

The name "CHAE GI-YEONG" also contains syllabic components, and "/甯/ 小 姐" are, too, depending on where the name is used. However, since these names are of CJK origin, rather than giving them a generic syllable reference, we will label the pieces by **CN**:

[NAME [SNP [CN CHAE]]

[GNP [CN GI] [- -] [CN YEONG]]], and

[NAME [SNP [CN甯]] [GNP [CN小] [CN姐]]].

3.10. Namesakes

Another potential place for observing multiple name pieces which could be treated as a single unit are namesakes. Personal names involving namesakes are "Benjamin Franklin /Smith/," "George Washington /Martinez/," or "Charles Jean Baptiste /Thiery/". That is, these names absorb the name of another famous individual. It is unclear at this point whether namesakes need to be tracked, so we include them just in case. There is evidence that such names may behave differently when one considers that the nickname "Biff" conflates with "Benjamin Franklin." Moreover, in some time frames, if the first name of an individual is "Benjamin," there is a significantly higher than expected probability that the middle initial will be "F."

We have been using the construct **NAME** up until this point without actually definition. The NAME construct is to a personal name what the "S" construct was for a sentence – an encapsulation of an entire idea. For namesakes, we will allow an inner phrase of NAME to be embedded in a larger NAME unit. More concretely, we will parse "Benjamin Franklin /Smith/ as if we were first parsing "Benjamin Franklin" and then embedding it in a larger unit:

[NAME [NAME [GN Benjamin] [FND Franklin]] [SNP [FN [Smith]]].

3.11 Occupations

It is well understood that surnames like "Baker," "Smith," and "Cooper" came from professions. Although the origin of such names is interesting, we do not view a name's origin as a commonly-needed component in a genealogical treebank.

However, there are names that are represented in the data where an understanding of occupation is required in order to render an adequate parse of the data. Consider the following names:

Dr. George Hodgson /Higgins/ (Physician)

(Contractor Architect) Frederich Albert /Telschow/

Hedwig /Dirksen/ Hausfrau

The phrases "Physician," "Contractor Architect," and "Hausfrau" are all occupations. We create two tags to accommodate these issues:

OCC = Occupation (other than a title)

OCCP = Occupational phrase.

This allows us to parse the occupational components of the above names into, respectively,

[OCCP [-LRB- -LRB-] [OCC Physician] [-RRB -RRB-]],

[OCCP [-LRB- -LRB-]

[OCC Contractor] [OCC Architect]

[-RRB -RRB-]], and

[OCCP [OCC Hausfrau]].

3.12 Descriptions and Attributes

The last of the components that we have observed as being required in name parsing have to do with descriptors and attributes. The following names include descriptors:

Alfred "The Great" King of /England/,

Cloderic "The Parricide" King Of /Cologne/, *and*

/Skjold/ King of Danes.

The descriptors here are “Great”, “Parricide,” and “Danes.” Note that each of these words, if processed through a normal part of speech tagger, would be given a different part of speech. “Great” is an adjective, and as seen before, would be given a part of speech, JJ. “Parricide” is a noun and would be given NN. “Danes” is a proper noun and would be called NNP. The “Danes” case is special since it conveys information about the race and location of the individual, and also, because it modifies the word “King” whereas the other pieces modify the name.

We introduce two additional constructs for handling attributes:

ATTP = Attributive Phrase

NNPD = Proper noun that is a demonym (such as English, Dane, French, Thane, etc.).

Using these new constructs, we can parse the descriptors for the example names as

[ATTP [“ “] [ADJP [DT The] [JJ Great]] [” ”]],

[ATTP [“ “] [NP [DT The] [NN Parricide]] [” ”]],

[PPTP [IN of] [NP [NNPD Danes]]].

We mentioned previously that translations of Native American names could be represented as sequences of NNPs embedded in a NAME. However, many of these names themselves use descriptors. They also frequently refer to animals, colors, and status. We will add some discrimination to NNPs and JJs to account for these special differences:

NNPA = Noun or Proper Noun describing an animal

JJC = Adjective describing a color (eg., black, red,…)

JJST = Adjective describing status (eg., sitting).

These additional constructs allow us to parse description-names such as

“/Black Bear/” = [NAME [GNP[JJC Black] [NNPA Bear]]]

“Sitting Bull” =[NAME [GNP[JJST Sitting] [NNPA Bull]]]

“George Big Deer” =

[NAME [GNP [GN George]][GNP [JJ Big] [NNPA Deer]]].

3.13 Generic Names?

We will create one more name construct. We do not know whether this construct is superfluous or if it has value, but we will define it in that event that it may turn out to be valuable. We will add **GNG** as a construct to account for generic nicknames such as “Buck,” “Buzz,” “Slim,” “Ace,” and so forth. Regular nicknames, like “Pat” for “Patrick” or “Patricia”; or “Johnny” for “Johnathan,” conflate with a particular subset of names. Generic nicknames, though, are not conflations for particular given names. Therefore, they have different properties and perhaps behave differently.

3.14 Constructs for Non-names as Names

We have identified numerous constructs associated with personal names. However, not every entry identified in a genealogical database as a personal name is indeed a name, and the same holds for name queries. Examples of non-names in Common Pedigree are “26 en 1913” and “Baby Boy.” For non-names or under-specified names, we use both name and generic parsing constructs to tag the constituents, and then wrap the full phrase in **NONNAME**.

4 THE PROCESS FOR DEVELOPING PNTB

As stated originally, the PNTB is a work in progress and is being created in a three-stage process. These stages are:

Stage 1: Pilot Parsing Experiment,

Stage 2: Personal Name Part of Speech Tagging, and

Stage 3: Conversion of Tags to Full Parses.

At the time of this paper, Stage 1 is complete, and Stage 2 is nearing the final stages of completion, and we are still early into Stage 3. We believe that for name parsing, Stage 2 is the most time-costly of the three stages, and this will be explained later. We will describe the processes for each of these stages and provide the status of each.

4.1 Pilot Parsing Experiment

In the first stage of PNTB creation, our goal was to identify what constructs would be required for parsing names and what issues would come up when creating a treebank. This initial phase of PNTB creation we will refer to as the “pilot parsing experiment” and we describe it briefly.

4.1.1. Pilot Parsing Name Selection

To get a clearer understand of PNTB creation issues, we thought it might be prudent to start with parsing a 5K set of personal names. We drew 10,000 Latin-script names from a 10M-entry name matching corpus (see [5]) and used half of these for creating the pilot parsed data set. The name matching corpus from which these names were drawn consists of names from Common Pedigree, from historical record collections, and Wikipedia.

4.1.2. Three-class Name Categorization

Next, we applied an existing rule-based tagger to these parses who sole responsibility was to mark each name piece with “FNF,” “GN,” or “O” for Father’s Family Name, Given Name, and Other. For specific examples:

Mary /Jones/	**GN FNF**
Samuel /Smith/	**GN FNF**
Maria del Sol Rodgrigues Martinez	**GN O GN GN FNF**
Harry N /Crane	**GN GN FNF**
Fredrik /Petterson/	**GN FNF**

The items that are marked in green are properly classified, whereas those marked in red are incorrect.

We reviewed these 5K three-way classifications. We found that when the personal name was pre-tagged with a patron judgment indicating the surname(s), only 8.52% of the automatic three-way tags were incorrect. When there was no pre-identified surname, the three-way tags were 10.44% in error.

4.1.3. From Categorization to Parse

As we moved in the pilot from name piece categorization to a full parse, if the name categorization was labeled as "correct," we used the initial three-way tags to provide a quick estimation of what the parses should look like. For example, "Harry N /Crane/" which was tagged as GN GN FNF would be marked as

[NAME [GNP [GN Harry] [GN N]] [SNP [FNF Crane]]].

Any entries whose categorizations were marked as erroneous were then parsed by hand.

4.1.4. Lessons Learned from Pilot

One of the "lessons learned" from this pilot is that a vast number of personal names fall into a small number of classes. Hence, to create a comprehensive PNTB, it would probably be beneficial to identify the different kinds of phenomena that occur in names rather than just randomly sampling data.

We also realized that a more comprehensive representation of the individual name pieces would be in order. For example, although it may be acceptable to merely treat the "N" in Harry N /Crane/ as a "GN," it might be better to categorize it as an initial, "ABBRI." Moreover, rather than treating a given name piece as part of the "other" class, it would be useful to categorize the name piece's role.

We also discovered that generating parses from personal name parts of speech is significantly less complex that the process of generating linguistic parses from linguistic parts of speech. However, a caveat moving forward would be that as the name tagset becomes more complex, the conversion from tags to parses will probably also become more complex.

A last lesson was that not all parsing errors affect genealogical search. If we drill down deeper into the 8.52% "errors," we found that 2.02% were associated with patronymics (which may or may not be in error as explained in Section 3.2); 0.72% were associated with multiple family names; 0.54% were associated with faulty handling of name particles; and 0.10% were mishandled honorifics. Depending on how a genealogical search system works, these cumulative 3.38% errors may have little bearing on search performance. The remaining 5.16% errors, however, could contribute adversely to search.

4.2. From Name Strings to PNPOS

By leveraging the lessons learned from the pilot, we were able to create the parsing constructs which we described in Section 3 and we were able to opt for a different strategy for creation of the whole PNTB. In particular, rather than grabbing a random set of names, which more than likely would fail to discover many of the interesting name patterns, we would iteratively process large portions of data and attempt to find interesting name phenomena. Following this stage, we would tag personal names based on the observed personal name parts of speech (PNPOS).

4.2.1. Finding Initial "Interesting" Names

To find numerous interesting name patterns, we desired to create a rule-based Personal Name Part of Speech Tagger which could try to identify the roles of each of the name pieces of every personal name (and sometimes, it identifies subpieces). Yet in order to create this rule-based system, we would need example interesting seed names.

To find these seeds, we sampled every 100th name of the previously-mentioned 10M-entry name matching corpus. We perused these by hand attempting to find as many types of name constructs as we could do manually. We considered the number of tokens in each name and, if it looked as if the name of that word count and structure would yield a novel parse, we stored the name.

4.2.2. Development of PNPOS tagging rules

Given these seed names, we created a rule-based system that could correctly PNPOS tag each of the seeds. This initial system contained over 300 rules. We also introduced into the system some constructs that were not referred to in Section 3 but which would hope to facilitate eventual parsing (such as **SS** for Spanish-like Surname, and **NAMESAK** for handling name sakes). The rule system was provided with knowledge of various kinds of particles; patronymic constructs; multilingual prepositions and determiners; titles; honorifics; occupations; locations; attributes; forms of the word "Maria"; and so forth.

We next applied this rule-based tagger to the entirety of the 10M name corpus. We reviewed the results of the parsing system and identified hundreds of additional rules necessary to add to the system. In fact, we coerced the system to predict rules that it would have preferred to have seen based on the word structure, and, when those rules seemed appropriate, they were added to the next instantiation of the tagger.

Afterward, we ran the system against the entirety of the unique Common Pedigree names and mined the output looking for novel rules that could be added to subsequent system iterations. We iterated on this process five additional times. We were careful to look not only at responses that were in Latin script, but also, where possible, at those that were non-Latin. Through this iterative process, we have thus far created 3863 different rules for parsing names.

4.2.3. Human Vetting of PNPOS

To begin assembling the final set of PNPOS tags, we applied the updated rule-predicting system to the set of all Common Pedigree names beginning with a letter between A-D. This set consists of 30.1M unique personal names. We will be repeating this process for the remainder of the corpus before we leave the PNPOS-creation stage, but we have not achieved this feat as of yet.

Before proceeding, we need to provide a definition. We will say that two personal names are in the same *PNPOS-tagged name class* if they both would be tagged with exactly the same set of tags (or, in other words, if the same PNPOS tagging rule applies accurately to both). Although there are some classes consisting of millions of names and others consisting of only thousands or hundreds, we wanted (at least, initially) to get a good sampling of what kinds of names correspond to the various kinds of name classes. Therefore, we selected no more than 100 names beginning with each of the letters A, B, C, and D for each name class for presentation to a human annotator.

We also created a vetting tool where a human could look at 35 names at a time with their corresponding rule and could mark each name's tag set as either correct ("YES"), incorrect ("NO"), probably correct/handle later ("Y-HL"), or probably incorrect/handle later.

At the time of writing this paper, the human judge had looked at 162,214 unique personal names. The human marked the proposed PNPOS-tagged names as:

145,830	Yes	4,453	Y-HL
1,825	No	10,106	N-HL.

If we take these Yes and Y-HL votes as both "sufficiently correct," we have **150,283** personal names tagged for PNPOS. These fall into 679 different PNPOS name classes. (It should also be commented that not all "N-HLs" are incorrect as they stand, but the human preferred that a new rule be created to handle those names more exactly.)

4.3. From PNPOS to Full Parses

Although we are currently at the initial stages of full parsing, we believe, based on the pilot experiment, that many of the full parses will be readily derivable from the PNPOS tags. Moreover, in the next section, we identify the frequency with which rules fire and find that many of the names have only one to four names which, after PNPOS tagging, should be straightforward to convert into parses. We will be using the same vetting tool to handle the parses as we had used for the PNPOS tagging. I

5 OBSERVATIONS

In Section 4.2, we mentioned that over 150,000 names have been tagged by a human with the first stages of parsing. It was also mentioned that the 150K represented 679 different name classes, and no more than 400 name examples were used for any one name class. Yet, when we described the pilot experiment in Section 4.1, it seemed that some name classes should be voluminous and others should be sparsely populated. The reader, therefore, may be interested in getting an estimate of the set size of each name class.

Since the 150K were drawn from the set of names whose first letter begins with A through D, we will show statistics on that same set of 30.1M unique names here.

Table 1 shows the top 50 rules as described in Section 4.2 that were observed when applied to this name set and the sizes of their name classes if all rules were correct.

Table 1: *Top Rules Identified on A-D Data Set*

Class Size	Rule	Class Size	Rule
6678282	G1 /S1/	84536	G1 /G2/
6394495	G1 S1	83781	I1 I2 S1
2882259	G1 G2 S1	81758	G1. /S1/
2840377	G1 G2 /S1/	77781	G1. S1
721987	G1 I1 /S1/	77002	S1 S2
713050	G1 I1 S1	76063	S1 /S2/
439980	G1 /P1/	74532	G1 pp1 SS1
438620	G1 SS1 SS2	71556	G1 P1 /S1/
429946	G1 P1	66658	G1 /p1S1bu/
324059	G1 G2 /P1/	66512	G1 p1S1bu
321204	G1 G2 P1	66432	G1 SS1 S1
282245	G1 S1 S2	65284	G1 /pp1 SS1/
270885	G1 G2 G3 /S1/	60549	G1 S1 SS1
267066	G1 G2 G3 S1	58605	G1 pP1 S1
265247	G1 U1	57853	G1 S1 CCD1 S2
255650	G1 pp1 S1	57402	G1 /S1 S2/
188258	G1 /pp1 S1/	56625	G1 G2 S1 S2
181463	G1 G2 SS1 SS2	54881	G1 G2 U1
158554	G1 G2	54613	U1 S1
152191	G1 S1 /S2/	52394	G1 pp1 /S1/
141981	G1 P1 S1	49739	G1 /P1 S1/
138112	NO PATTERN	49084	G1 pp1 pd1 S1
135861	G1 /SS1 SS2/	48602	G1 I1 /P1/
86653	G1 G2 pp1 S1	47956	G1 I1 P1
86323	S1	46441	G1 /p1S1b/
86037	I1 I2 /S1/	46118	G1 p1S1b

The rules from Table 1 represent 26.42M (or 87.7%) of all the names that occur in the A-D data set. Note that one of these slots is "No Pattern" where no rules fired for the particular names. This means that approximately 12.5% of personal names do not fit these rules. Although 12.5% seems like a small number, Common Pedigree is fast approaching 1B names, which means that about 125M personal names would not fall into the top 50 classes.

The remaining 12.5% fall into a long list of rule types. Table 2 (see the next page) shows each successive set of 50 additional rules and the number of unique names that were identified with each of those rules. It then shows the cumulative percentage of names that would have been parsed by the time all of those rules had been applied.

Table 2: *Cumulative Number of Processed Names by Rule*

Ordered 50-Rule Sets	**Number of Names Handled**	**Cumulative Percentage of Names Handled**
Rules 1-50	26,420,983	87.7%
Rules 51-100	1,325,206	92.2%
Rules 101-150	642,160	94.3%
Rules 151-200	347,753	95.5%
Rules 201-250	232,893	96.2%
Rules 251-300	173,128	96.8%
Rules 301-350	128,270	97.2%
Rules 351-400	97,249	97.6%
Rules 401-450	80,389	97.8%
Rules 451-500	68,415	98.1%

Note that even after 500 rules have been applied, there are still about 2% of the personal names upon which no rule fired, which would equate to 20M names from the full Common Pedigree. This provides strong evidence of the value of a Personal Name Treebank when it is applied to the handling of genealogical search on a very temporally and linguistically diverse corpus such as Common Pedigree.

6 CONCLUSIONS AND FUTURE PLANS

As we look forward into the future, there is obviously a continuing need on our part to grow the number of personal name parts of speech to cover the remaining Common Pedigree names (outside of those beginning with A-D) and then wrap these in parse structure. We expect this process to take several months.

Upon its completion, we will be performing two studies. The first of these is to see how well existing, trainable, constituency parses can learn the constructs of the PNTB. This process may lead to additional refinements of the PNTB.

Afterwards, we plan to see if these automatic parses can be used as a filter for automatic name matching (see [5] for extended details). Our hope is that, once a genealogical search is performed, the search engine can parse the query and the result set and ensure that each result has a name syntax which is consistent with the kind of name that was in the query.

Lastly, we will be seeking to make the PNTB releasable to the world at large. It is our expectation that this PNTB will be of tremendous value to the greater research community for identifying how to process human names.

REFERENCES

[1] M. Marcus, B. Santorini, M. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguistics*, 19(2), 1993.

[2] Wikipedia, "Treebank," Definition from 01/24/2012.

[3] E. Charniak "A Maximum-Entropy-Inspired Parser," NAACL'00, pp. 132-139

[4] M. Collins, "Head-driven Statistical Models for Natural Language Parsing", PhD Thesis, 1999

[5] P.Schone, C. Cummings, S. Davey, M. Jones, B. Nay, M. Ward, "Comprehensive Evaluation of Name Matching Across Historic and Linguistic Boundaries," FHTW@RootsTech2012, to appear.