PAGC Geocoding Topics

Revised 2008-11-30 for **Pagc-0.2.0.**

Contents

THE REFERENCE SHAPESET

The reference data must be in the form of a shapeset. A shapeset will be the set of three files, named REFERENCE_SHAPESET_NAME.shx, REFERENCE_SHAPESET_NAME.shp and REFERENCE_SHAPESET_NAME.dbf. The first two files will contain the positional data of the shapes, whereas the third, an xbase table, is the reference attribute table. It will contain the streetnames, address ranges and other data. The shapeset, a file format in the public domain, is a file that can be read and produced by many GIS programs.

The coordinates used in the shapefile are assumed by the program to be in decimal degrees.

In the build phase, however, it is only the shapeset's xbase attribute table that is used by the program. The postal address related information is extracted from this file, indexed, and standardized in preparation for matching against user attribute files.

Reference Attribute Table Data Values

The reference table data is assumed to be represented in 7 bit ASCII characters. The latin-1 characters that may exist in some schemas are modified, for standardization purposes, to remove diacritical marks and accents.

In Statistic Canada schemas, if there exists an ARC_GROUP field for a record and the value does not begin with the letter "A", it is bypassed as a non-addressable feature. In all other schemas, including newer Statscan RNF files, a record is bypassed only if it has a blank street name field.

In every field (except address range fields) a field is regarded blank if it is in fact blank or if it begins with an underscore.

HOUSE

> In fields which represent part of an address range a civic number with the value of 0 is interpreted literally, except when using a Statistics Canada schema, in which case it is intrepreted as a blank. Non numeric characters are ignored. The first string of digits encountered is interpreted as the civic number. If a fraction is included, the number is rounded up. Any initial fraction is rounded up to 1. Milepost, coordinate-style and addresses that depend on box numbers and rural routes or post office boxes cannot be handled by this version.

CITY, PROV, POSTAL, NATION

> If a reference field of this type has a value for a block face, but there is no corresponding address range, it is omitted from the standardized reference.

Standardization of a Reference Record

The unstandardized reference record is parsed, according to the schema, into a MICRO, a MACRO left and a MACRO right portion and each portion is standardized separately. The AddressRange Numbers are omitted from the MICRO portion when it is sent to the standardizer.

The standardizer uses only rules of the ARC_C class (See rule types) to standardize the MICRO portion of the record. These rules differ from the MICRO_C class in that they do not possess HOUSE output symbols. That is, a MICRO_C rule is like a CIVIC_C and an ARC_C rule combined.

The standardizer creates multiple, ranked standardizations of the MICRO portion of the record. If no standardization can be found, an error message is logged and the record is bypassed. Otherwise, each standardization is examined to determine how closely it corresponds to the unstandardizd data. In particular, a standardization is downgraded if an attribute present in the reference data is missing (except for those attributes not explicitly included in the reference schema) or an attribute missing in the reference data pops up in the standardization. The best standardization with none of these kinds of errors is

accepted. Otherwise, an error message is logged, and the best standardization with the fewest of these errors is accepted. The standardization used in such a case is also error logged.

The standardization accepted may produce attributes not present in the reference schema. Some schemas are rather sparse in attributes. These extra-schema attributes will be dealt with in the match phase by the redirection strategy (See Redirection).

The left and right sides of the MACRO (See Micro/Macro) are first examined to see if the feature is a MACRO boundary - i.e. if they differ. If they don't then only one side is standardized. If a blockface MACRO, stipulated by the schema, is absent even though an address range exists for that blockface, an error is logged. If no standardization is found, then an error is logged and the record bypassed. The standardizer uses only rules of the MACRO_C class to standardize these portions of the record.

Reference Attribute Schema.

There can be many different representations of postal address attribute data in reference attribute tables. **PAGC** must be able to determine this representation before it can standardize and index. It can either attempt to detemine the schema by probing the field names in the reference attribute table or the user can provide it with the schema beforehand. The program needs to know the postal attribute (See Postal Attributes) associated with each field and the comparison type (See Comparison Types). It does not need the BLDNG, UNITH, UNITT, BOXH, BOXT, RR or UNKNWN attributes and these should not be included in schemas even if present in the reference.

If no schema table is provided on the command line with the **-s** switch, the program attempts to discover the the schema from the reference table's field names. The program first probes the reference attribute table for either the US Tigerline format (by looking for a "FEDIRP" field) or a Statistics Canada format (by looking for a "ADD_FM_LE" field). If either of these fields are discovered, it probes for additional address fields in case these have been added.

The case (upper, lower, mixed) of the field names is not important, nor the order in which they appear in the file.

Field names not part of a schema or on any of the probe lists are ignored. Reference Attribute tables often contain data that is irrelevant to postal address geocoding.

Built-in Schemas

Certain entire schemas are built-in to the program. In particular:

Tiger

PAGC is set up to recognize the US Census Tiger Line format in the reference attribute table without a schema table. It looks for the address range fields *FRADDL ,TOADDL, FRADDR, TOADDR* and interprets them as the NUMBER_INTERVAL_LEFT_RIGHT comparison type. It also looks for the CHAR_SINGLE fields *FEDIRP, FNAME, FEDIRS, FETYPE*, corresponding to the *PREDIR, STREET, SUFDIR, SUFTYP* attributes respectively. It also looks for the POSTAL_LEFT_RIGHT fields *ZIPL, ZIPR*.

Statscan

PAGC is set up to recognize the Statistics Canada road network format. It looks for the address range fields *ADDR_FM_LE, ADDR_TO_LE, ADDR_FM_RG, ADDR_TO_RG* and the CHAR_SINGLE fields *DIRECTION, NAME, TYPE*, corresponding to the *PREDIR, STREET, SUFTYPE* attributes. Note that it has no POSTAL or indeed any other MACRO attribute. Note too that the DIRECTION and TYPE fields are actually ambiguous. A SUFTYPE in Anglophone Canada may be more reasonably assumed to be a PRETYPE in Francophone Canada.

Because of the possibility that both of these formats can be extended by adding additional fields (a CITY field, for example, or POSTAL_LEFT_RIGHT fields for Statscan), the program will also probe for field names for attributes absent from the schema.

Reference Field Names

The field names (with attribute and comparison types) that can be recognized in reference attribute tables by field name probing are listed here. It is an error to have more than one field name set from the same attribute.

If you wish your reference attribute table to be recognized properly by **PAGC**, you can always change your xbase field names to correspond.

Beware of the non-address field names that may conflict with field names **PAGC** may recognize. In particular, *NAME* is intrepeted as a CHAR_SINGLE STREET name. You will need a schema table if such conflicts exist.

- HOUSE field names. The field names used for the HOUSE attribute, arranged by comparison type, are as follows.
    - NUMBER_INTERVAL_LEFT_RIGHT. This is the most common type, used by the Statistics Canada and US Tigerline distributions. The address range for the left blockface precedes the address range for the right blockface.
        - FRADDL ,TOADDL, FRADDR, TOADDR
        - ADDR_FM_LE, ADDR_TO_LE, ADDR_FM_RG, ADDR_TO_RG
        - ADD_FM_LEFT ,ADD_TO_LEFT, ADD_FM_RIGHT, ADD_TO_RIGHT

        - FROMLEFT ,TOLEFT, FROMRIGHT, TORIGHT
        - L-F-ADD ,L-T-ADD, R-F-ADD, R-T-ADD
        - L_F_ADD ,L_T_ADD, R_F_ADD, R_T_ADD
        - LEFTADD1 ,LEFTADD2, RGTADD1, RGTADD2
        - INITIAL_LEFT_NUMBER ,FINAL_LEFT_NUMBER, INITIAL_RIGHT_NUMBER, FINAL_RIGHT_NUMBER
        - L-ADD.FROM ,L-ADD.TO, R-ADD.FROM, R-ADD.TO
        - L-ADD_FROM ,L-ADD_TO, R-ADD_FROM, R-ADD_TO
        - L_ADD_FROM ,L_ADD_TO, R_ADD_FROM, R_ADD_TO
        - L_ADD.FROM ,L_ADD.TO, R_ADD.FROM, R_ADD.TO
        - L_ADD_FROM ,L_ADD_TO, R_ADD_FROM, R_ADD_TO
        - L_ADD-FROM ,L_ADD-TO, R_ADD-FROM, R_ADD-TO
        - LADD.FROM ,LADD.TO, RADD.FROM, RADD.TO
        - LADD_FROM ,LADD_TO, RADD_FROM, RADD_TO
        - LADD-FROM ,LADD-TO, RADD-FROM, RADD-TO
        - LADD.FR ,LADD.TO, RADD.FR, RADD.TO
        - LADD_FR ,LADD_TO, RADD_FR, RADD_TO
        - LADD-FR ,LADD-TO, RADD-FR, RADD-TO
    - NUMBER_INTERVAL. This is a single address range style. Blockfaces are not differentiated.
        - FR.ADD , TO.ADD
        - FR_ADD , TO_ADD
        - FR-ADD , TO-ADD
        - F.ADD , T.ADD
        - F_ADD , T_ADD
        - F-ADD , T-ADD
        - LOW , HIGH
        - ADD.FROM , ADD.TO
        - ADD_FROM , ADD_TO
        - ADD-FROM , ADD-TO
        - FROM ,TO
        - FM ,TO
        - FR ,TO
        - ADD1 ,ADD2
    - NUMBER_SINGLE HOUSE. This non-address range type is not yet fully supported in this version.
        - HOUSE
        - CIVIC
        - ADDRESSNUMBER
        - COMPLETEADDRESSNUMBER
        - HOUSENUM
        - NUMBER
        - HOUSE.NUM
        - HOUSE_NUM
        - HOUSE-NUM
        - HOUSE.NUMBER

        - HOUSE_NUMBER
        - HOUSE-NUMBER
        - HOUSE_NUMB
        - ADD
- CHAR_SINGLE MICRO ATTRIBUTES.

  It is assumed that non-HOUSE MICRO attributes will be using the CHAR_SINGLE comparison type.

    - PREDIR
        - FEDIRP
        - DIRECTION
        - PREDIR
        - PREDIRECTIONAL
        - DIR
        - PREFIX
        - FDPRE
        - PRE.DIR
        - PRE_DIR
        - PRE-DIR
        - STREET.DIR
        - STREET_DIR
        - STREET-DIR
    - PRETYP
        - PRETYP
        - PRETYPE
        - STREETPREFIXTYPE
        - PRE.TYPE
        - PRE_TYPE
    - STREET
        - STREET
        - STREETNAME
        - FNAME
        - STREET.NAME
        - STREET_NAME
        - STREET-NAME
        - STREET_NAM
        - ST.NAME
        - ST_NAME
        - ST-NAME
        - STR.NAME
        - STR_NAME
        - STR_NAME
        - NAME
    - SUFTYP
        - FETYPE

        - SUFTYP
        - SUFTYPE
        - STREETPOSTTYPE
        - STREETTYPE
        - TYPE
        - FTYPE
        - STREET.TYPE
        - STREET_TYPE
        - STREET-TYPE
        - STREET_TYP
        - ST.TYPE
        - ST_TYPE
        - ST-TYPE
        - STR.TYPE
        - STR_TYPE
        - STR-TYPE
    - SUFDIR
        - SUFFIX
        - FEDIRS
        - FDSUF
        - POSTDIRECTIONAL
        - SUFFIXDIR
        - SUFDIR
        - SUF.DIR
        - SUF_DIR
        - SUF-DIR
        - SUFFIX.DIR
        - SUFFIX_DIR
        - STREET.DIR
        - STREET_DIR
        - STREET-DIR
- Non-postal MACRO field names

  Use of a schema table is necessary for reference shapesets that span state/provincial or national boundaries, and that possess addressable features that straddle the boundaries.

    - CITY (CHAR_LEFT_RIGHT)
        - LEFT_MUN, RIGHT_MUN
        - LEFT_CITY, RIGHT_CITY
        - LEFT_PLACE, RIGHT_PLACE
        - LCITY, RCITY
        - L.CITY, R.CITY
        - L_CITY, R_CITY
        - L-CITY, R-CITY
        - CITYL, CITYR

        - CITY_L, CITY_R
        - CITY-L, CITY-R
        - CITY.L, CITY.R
    - CITY (CHAR_SINGLE)
        - CITY
        - PLACENAME
        - LOCALITY
        - MUNICIPAL
    - PROV (CHAR_SINGLE)
        - PROV
        - STATE
        - REGION
    - NATION (CHAR_SINGLE)
        - COUNTRY
        - NATION
- Postal field names

  The postal field names recognized are as follows, arranged by comparison type.

    - POSTAL_LEFT_RIGHT_SPLIT
        - LZIP, LZIP4 RZIP, RZIP4
        - ZIPL, ZIP4L,ZIPR, ZIP4R
        - ZIP.LEFT, ZIP4.LEFT,ZIP.RIGHT, ZIP4.RIGHT
        - ZIP_LEFT, ZIP4_LEFT,ZIP_RIGHT, ZIP4_RIGHT
        - ZIP-LEFT, ZIP4-LEFT,ZIP-RIGHT, ZIP4-RIGHT
        - LEFT.ZIP, LEFT.ZIP4,RIGHT.ZIP, RIGHT.ZIP4
        - LEFT_ZIP, LEFT_ZIP4, RIGHT_ZIP, RIGHT_ZIP4
        - LEFT-ZIP, LEFT-ZIP4,RIGHT-ZIP, RIGHT-ZIP4
        - L.ZIP, L.ZIP4,R.ZIP, R.ZIP4
        - L_ZIP, L_ZIP4,R_ZIP, R_ZIP4
        - L-ZIP, L-ZIP4,R-ZIP, R-ZIP4
    - POSTAL_SPLIT
        - ZIP, ZIP4
        - ZIP, PLUS4
        - FSA, LDU
        - ZIPCODE, ZIP4
        - ZIP-CODE, ZIP4
        - ZIP_CODE, ZIP4
    - POSTAL_LEFT_RIGHT
        - LEFTZONE, RIGHTZONE
        - LEFT_ZONE, RIGHT_ZONE
        - LZIP, RZIP
        - ZIPL, ZIPR
        - ZIP.LEFT, ZIP.RIGHT
        - ZIP_LEFT, ZIP_RIGHT
        - ZIP-LEFT, ZIP-RIGHT

        - LEFT.ZIP, RIGHT.ZIP
        - LEFT_ZIP, RIGHT_ZIP
        - LEFT-ZIP, RIGHT-ZIP
        - LEFT_ZIP_CODE, RIGHT_ZIP_CODE
        - L.ZIP, R.ZIP
        - L_ZIP, R_ZIP
        - L-ZIP, R-ZIP
        - L_PCODE, R_PCODE
        - PCODE_L, PCODE_R
        - PCODE_LE, PCODE_RG
        - PCODE_LEFT, PCODE_RIGHT
        - LEFT_FSA, RIGHT_FSA
        - PC_L, PC_R
        - POSTAL_LEFT, POSTAL_RIGHT
        - CODE_LEFT, CODE_RIGHT
        - PC_LEFT, PC_RIGHT
    - POSTAL_SINGLE
        - ZIP
        - ZIPCODE
        - ZIP-CODE
        - ZIP_CODE
        - ZONE
        - PCODE
        - POSTAL
        - POSTALCODE
        - POSTAL_CODE
        - POSTAL-CODE
        - FSA
        - PC
        - CODE

Schema Xbase File

The schema file is an xbase file that describes the schema of the reference shapeset. If the program is unable to discover the schema of the reference attribute table by probing the field names, it will be necessary for the user to provide the program with the switch **-s**SCHEMA_TABLE_NAME along with the **-b** switch. The xbase (.dbf) table consists of six to eight columns. The extension (.dbf) need not be provided. It will have as many rows as the number of active postal attributes that are incorporated.

Schema field structure

The schema xbase (dbf) file consists of six fields:

Field Number	Field Name	Type	Size
1	ATTRIB	C	8
2	COMPARE	C	35
3	NAME1	C	25
4	NAME2	C	25
5	NAME3	C	25
6	NAME4	C	25

The Structure of a Schema Table

An additional two fields can also, if desired, be appended for the purposes of overriding the default weights (See the default matching weights):

Field Number	Field Name	Type	Size
7	M	F	10
8	U	F	10

Optional match weight fields

The Attrib Field

The Attrib Field. In the schema table there will be one row for each active postal attribute (See Postal Attributes). In other words, the postal attribute is the key for the row: there is one attribute per row, one row per attribute. The HOUSE attribute is used, for example, for the fields associated with address ranges, and the POSTAL attribute is used for zip/postal codes, STREET for the streetname (such as *Tenth* in *West Tenth Ave* , PREDIR for a predirectional (such as *West* in *West Tenth Ave*, SUFTYP for the posttype (such as *Ave* in *West Tenth Ave*, etc.

The Compare Field

Compare Field. With each attribute there is associated a comparison type. Each comparison type specifies the method by which the relevant fields in the reference record will be compared to the corresponding elements of the user's address record. The comparison types that **PAGC** defines are listed below.

Comparison Types

NO_COMPARISON

This comparison type is not used in schema tables. It is used internally for redirection (See Redirection).

CHAR_SINGLE

The reference and target fields are compared by matching the character string letter for letter. CHAR_SINGLE comparisons may also be extended to string similarity comparisons. This is the most common comparison type for fields other than POSTAL or HOUSE attribute fields. PRETYPE, PREDIR, QUALIF, STREET, SUFDIR, and SUFTYPE will ordinarily use this comparison type.

CHAR_LEFT_RIGHT

The reference has a left and a right fields, either of which can be matched to the target. This is for MACRO fields such as CITY that may be different for the left and right blockfaces of a street. String similarity measures may be used here too.

NUMBER_SINGLE

The reference and target match on a single number.

NUMBER_INTERVAL

The target number must fall between the two numbers (*from* and *to*) in the reference.

NUMBER_INTERVAL_LEFT_RIGHT

The reference has four numbers, a from-to interval on the left and one on the right. The target number must fall between either one of the intervals. This is the most common of the NUMBER comparison types used in postal address geocoding. This comparison type may also include a similarity comparison (transpositions only).

POSTAL_SINGLE

Reference and target match letter for letter on a single postal/zip code field.

POSTAL_SPLIT

This is the comparison type used when the reference splits the postal code into two separate fields (eg zip, zip4 or fsa, ldu).

POSTAL_LEFT_RIGHT

The comparison type used for postal code with both left and right fields, either of which can be matchted to the target. This is the most common POSTAL type used in postal address geocoding.

POSTAL_LEFT_RIGHT_SPLIT

This comparison type combines the previous two.

The Name Fields

The Four Name Fields. The four fields named NAME1, NAME2, NAME3 and NAME4 will give the fieldnames in which the components of the comparison type will be found in the reference file. The fieldnames, in the schema fields NAME1 .. NAME4 are expected in the precedences given below. In other words, the names should appear in the order specified even if that is not the order in which they appear in the record structure.

CHAR_LEFT_RIGHT and POSTAL_LEFT_RIGHT

LEFT > RIGHT

POSTAL_SPLIT

GENERAL > SPECIFIC (e.g. ZIP BEFORE ZIP+4)

NUMBER_INTERVAL

FROM > TO

NUMBER_INTERVAL_LEFT_RIGHT

FROMLEFT > TOLEFT > FROMRIGHT > TORIGHT

POSTAL_INTERVAL_LEFT_RIGHT

FROMLEFT > TOLEFT > FROMRIGHT > TORIGHT

POSTAL_LEFT_RIGHT_SPLIT

GENERAL_LEFT > SPECIFIC_LEFT > GENERAL_RIGHT > SPECIFIC RIGHT

The M and U Fields

The M and U Fields. You may also, if you wish, add fields to set the matching weights for each attribute. Add a field named M for the match weight and/or add a field named U for the mismatch weight. These are the weights used to weight the matches between the

user's address and the reference records and represent the probability of random false negatives and random false positives. If these fields are absent, the default values (See matching default values) for each attribute is used. If either field is present, but has a blank or 0.0 value for any attribute, that attribute will be given the default value. The default value is used, therefore, unless the field is present in the schema table and the non-blank value in the field is greater than 0.0 but less than 1.0.

Example Schema Table

Example: the schema file **freetig.dbf**, giving the free shapefile distributions of TigerLine files will have the following 6 records:

ATTRIB	COMPARE	NAME1	NAME2	NAME3	NAME 4
HOUSE	NUMBER_INTERVAL_LEFT_RIGHT	Fraddl	Toaddl	Fraddr	Toaddr
PREDIR	CHAR_SINGLE	Fedirp			
STREET	CHAR_SINGLE	Fename			
SUFTYP	CHAR_SINGLE	Fetype			
SUFDIR	CHAR_SINGLE	Fedirs			
POSTAL	POSTAL_LEFT_RIGHT	Zipl	Zipr		

freetig.dbf

ATTRIB	COMPARE	NAME1	NAME2	NAME3	NAME 4	M
HOUSE	NUMBER_INTERVAL_LEFT_RIGHT	Fraddl	Toaddl	Fraddr	Toaddr	.99
PREDIR	CHAR_SINGLE	Fedirp				
STREET	CHAR_SINGLE	Fename				
SUFTYP	CHAR_SINGLE	Fetype				
SUFDIR	CHAR_SINGLE	Fedirs				
POSTAL	POSTAL_LEFT_RIGHT	Zipl	Zipr			

The above table with an M field

NONATTRIBUTE AND FLAG SCHEMA LINES

These should be specified after the attribute lines, for nonattribute fields in the schema and for schema flags

NONATTRIBUTE SCHEMA LINES

XSTREET

This flag is entered in the ATTRIB column, with NO_COMPARISON in the COMPARISON column, and the two column names in the next two columns. (Example below). This directs PAGC to index the cross-street names for intersection lookup.

OCCUP1

This flag is entered in the ATTRIB column, with NO_COMPARISON in the COMPARISON column, and the column name of the occupancy field in the next column. This directs PAGC to include a single occupancy field in the standardized address record.

OCCUP2

This flag is the same as the OCCUP1, except it indicates two fields for the occupancy information.

COORDS

This flag is entered in the ATTRIB column, with NO_COMPARISON in the COMPARISON column, and the column name of the two coordinates fields in the succeeding column. This directs PAGC to use coordinates in the xbase file of the shapeset rather than the shapefile.

SOURCEID

This flag is entered in the ATTRIB column, with NO_COMPARISON in the COMPARISON column, and the column name of the source id field in the next column. This directs PAGC to retain designated field contents in each shape row as the source id for the address in that row.

ATTRIB	COMPARE	NAME1	NAME2	NAME3	NAME 4
XSTREET	NO_COMPARISON	F_XSTREET	T_XSTREET		

Example of a schema line for the XSTREET flag

SCHEMA FLAGS

Schema flags are specified in the ATTRIB column. The rest of the line is not used.

FLSTATS

This flag tells PAGC to build a statistics file

FLPSEUD

This flag tells PAGC to enable pseudo edits of reference records to assist in overcoming missing information in the reference

FLZBLNK

This flag tells PAGC that 0 in a HOUSE field means blank

FLRNFRE

This flag tells PAGC to redirect TYPES for RNF type schemas

FLCONPR

This flag tells PAGC to use the Berkeley Concurrent Private threading model

FLNOSEX

This flag tells PAGC not to stop collecting candidates upon finding a perfect match

FLRPSEQ

This flag tells PAGC to read a point shapefile sequentially rather than using the shx index file

FLOFFST

This flag tells PAGC to retain the unstandardized name of the base streetname and substitute it for the standardized name when outputting the field

TWO MORE SCHEMA TABLE EXAMPLES

ATTRIB	COMPARE	NAME1	NAME2	NAME3	NAME4
HOUSE	CHAR_SINGLE	BLDG_NUM			
PREDIR	CHAR_SINGLE	PREFIX_DIR			
STREET	CHAR_SINGLE	STREETNAME			
SUFTYP	CHAR_SINGLE	STREETTYPE			
SUFDIR	CHAR_SINGLE	SUFFIX_DIR			
CITY	CHAR_ALT	CITY	CITY_USPS		
POSTAL	POSTAL_SPLIT	ZIP	ZIP4		
OCCUP1	NO_COMPARISON	UNIT_INFO			
PRETYP	CHAR_SINGLE	PREFIXTYPE			
FLRPSEQ					
FLOFFST					
SOURCEID	NO_COMPARISON	PIN			

Example of a schema for a parcels dataset.

ATTRIB	COMPARE	NAME1	NAME2	NAME3	NAME4
HOUSE	NUMBER_INTERVAL_LEFT_RIGHT	L_F_ADD	L_T_ADD	R_F_ADD	R_T_ADD
PREDIR	CHAR_SINGLE	PREDIR			
STREET	CHAR_SINGLE	STREETNAME			
SUFTYP	CHAR_SINGLE	TYPE			
SUFDIR	CHAR_SINGLE	SUFDIR			
CITY	CHAR_LEFT_RIGHT	CITYLEFT	CITYRIGHT		
POSTAL	POSTAL_LEFT_RIGHT	ZIP5_L	ZIP5_R		
XSTREET	NO_COMPARISON	F_XSTREET	T_XSTREET		
SOURCEID	NO_COMPARISON	TLGID			
FLOFFST					

Example of a schema for a streets dataset.

Files Comprising a Normalized, Schematized Reference

REFERENCE.pgc

This file is produced by pagc_build_schema to pass the schema information to the matching process. Matching cannot proceed if it is absent. It is the last file created before the build process completes.

REFERENCE.pgx

This file contains the normalized data records. It is the main data file. It is a Berkeley b-tree file. The lookup name for each record is the original shapeset entity number for the record it represents.

REFERENCE.ix0

This file is the full street name index. It is a Berkeley b-tree file. A lookup by full street name (+/- postal code) returns the lookup name (same as the record number in the original shapeset) of the normalized record to which it corresponds.

REFERENCE.ix1

This file is the root street name index. It is a Berkeley b-tree file. A lookup by root street name (+/- postal code) returns the lookup name (same as the record number in the original shapeset) of the normalized record to which it corresponds.

REFERENCE.ix2

This file is the soundex street name index. It is a Berkeley b-tree file. A lookup by the soundex code of the root street name returns the lookup name (same as the record number in the original shapeset) of the normalized record to which it corresponds.

REFERENCE.ix3

This file is the approximate street name index. It is a paging trie that uses the Berkeley memory pool facility. A lookup by root street name returns a list of all the street names within maximum edit distance. Each element of the list is passed to the root street name index (*.ix1)

REFERENCE.ix4

This file is the point index. It is used to index the main data records by the start and end points of a blockrange. It is created only if the FLPSEUD flag is stipulated in the schema file. It is a Berkeley b-tree file. The lookup name is the latitude and longitude of a point

REFERENCE.ix5

This contains the variable length shape information for each record: the x and y coordinates for a point record or the set of x and y coordinates pairs for an arc record. It is a Berkeley b-tree file. The lookup name for each record is the original shapeset entity number for the record it represents.

REFERENCE.ix6

This contains the index of concatenated intersection names. It is created only if the XSTREET flag is used in the schema. It is a Berkeley b-tree file. The lookup name is a concatenation of the street and cross-street names and the value returned is the lookup name of the normalized pgx record.

REFERENCE.ix7

This contains the soundex index of concatenated intersection names. It is created only if the XSTREET flag is used in the schema. It is a Berkeley b-tree file. This corresponds to *.ix2, except the the lookup name is the concatenated name as used in *.ix6

REFERENCE.ix8

This contains the approximate index of concatenated intersection names. It is created only if the XSTREET flag is used in the schema. It is a paging trie that uses the Berkeley memory pool facility. This corresponds to *ix3 except the lookup is the concatenated name as used in *.ix6

build_log.err

This file contains the output of pagc_build_schema if the -l flag was given. In addition to warnings and errors it also documents the construction of the schema. In particular it records the postal attribute assignations, the flags, and the indices created. It records the source records that it cannot standardize and those it standardizes in a manner different from that implied by the fields of the source record.

REFERENCE.sts

This file contains the statistics on the standardization rules consulted and used in the normalization of the data.

__db.001 and __db.002

These are the memory cache files produced by Berkeley DB for the environment.

Scoring the Reference Records

The method of scoring candidates is a modified version of the standard Fellegi-Sunter method. As in that method, each attribute in the reference schema has two values associated, the match weight **m** and the mismatch weight **u**. A reference record is matched with the user's record and scored attribute by attribute, and each attribute contributes to the total a value determined by **m** and **u**. If the standardizations of the user and reference records agree on that attribute, a contribution to a total sum is calculated by log(m / u). If they don't agree, the contribution is log((1 - m) / (1 - u)). The total of the contributions from each schema attribute is the total score of the reference record candidate.

The Default Match and Mismatch Weights

The default match and mismatch weights that the program uses are as follows.

HOUSE

match = 0.999 mismatch = .05.

STREET

match = 0.9 mismatch = .01.

CITY

match = 0.9 mismatch = .1.

POSTAL

match = 0.9 mismatch = .1.

SUFDIR

match = 0.85 mismatch = .1.

SUFTYP

match = 0.85 mismatch = .1.

PREDIR

match = 0.8 mismatch = .1.

PRETYP

match = 0.7 mismatch = .1.

QUALIF

match = 0.7 mismatch = .1.

These default weights can be reset in the schema table. If you wish to change the defaults, you will need to use a schema table, even though the program would recognize your schema without one. The match weight is the probability that a match is not a random false positive and the mismatch weight is the probability that a mismatch is not a random false negative. In theory these values should be determined empirically, but in practice the default values appear to suffice.

Note : It is possible also to change these values after the reference dataset is built by amending the pgc file.

Scoring modifications

Agreement for a particular attribute is determined by the comparison type (See Comparison Types) associated with that attribute in the reference schema. However, the method of comparison demanded by the type is modified and elaborated in order to introduce similarity comparison and to handle attribute redirection. The specifics of these modifications follow.

CHAR_SINGLE

In fields of this comparison type, where there is no attribute re-direction, the user and reference values are first examined for an exact match. If there is an exact match, the match weight is applied. If not, the two strings are examined for similarity and given a similarity measure. This value is then interpolated into the interval between the match and mismatch weights to determine the value to add to the score.

POSTAL

The standardizer ensures that postal codes for the user and the reference are both valid foms. However, it may be that the codes of one is a different length than that of the other. One may, for instance have both the zip and zip+4 while the other may have just the zip. Whichever of the two is longer will be truncated, for the purposes of the comparison, to the length of the shorter. The two are first examined for an exact match. An exact match gets the match weight. If no exact match, they are examined for similarity. The similarity measure produced is then interpolated into the interval between the match and mismatch weights, and this value is added to the score. If the reference is using block faces, the left and right

postal codes may differ. Both will be compared to the user postal code and the better of the two scores will be taken.

HOUSE

House numbers are converted to integers and are examined to see, in the case of NUMBER_INTERVAL_LEFT_RIGHT, if they fall within the range of either the left or right block face. A success for either will result in the addition of the match weight. If neither, then simple transpositions are tested, and a similarity measure (determined by Jaro's algorithm) is used to interpolate into the range between the match and mismatch weights. Provision is also made for address parity. Parity is a common (although undoubtedly not universal) method of distinguishing opposite block faces. That is, the even numbers will be on one side of the street, and the odds on the other. A non-match on parity (if there are numbers on only one side, for example) will slightly decrease the score of an address range. However, a parity mismatch will be ignored in geocoding - a warning is issued in the error file and the address is geocoded as if the parity corresponded. The arithmetic proximity of an address range is also used in scoring. That is, the ranges closer to the target number will score higher. This is useful in selecting records to edit and to look for range continuations.

Redirected Attributes

The use of a redirection strategy (See Redirection) to deal with the problem of differing user and reference schemas complicates comparison of those attributes to which other attributes are redirected. It may be, for example, that when SUFDIRS are redirected to PREDIRs, that a SUFDIR in the user address (*10th Ave West*) corresponds exactly with a PREDIR in the reference (*West 10th Ave*), in which case we want to these values to be considered a match. However, suppose you have a name with both a PREDIR and a SUFDIR in the reference, say *West 10th Ave East*. The agreement on the *West* would count for nothing. The fewer attributes in the schema, the more acute this problem becomes. Therefore, a similarity measure is used when redirection occurs. If an exact match can't be obtained, then Jaro's algorithm is used to weight the number of common attributes and the order in which they appear.

Comparison Types

NO_COMPARISON

This comparison type is not used in schema tables. It is used internally for redirection (See Redirection).

CHAR_SINGLE

The reference and target fields are compared by matching the character string letter for letter. CHAR_SINGLE comparisons may also be extended to string similarity comparisons. This is the most common comparison type for fields other than POSTAL or HOUSE attribute fields. PRETYPE, PREDIR, QUALIF, STREET, SUFDIR, and SUFTYPE will ordinarily use this comparison type.

CHAR_ALT

The reference has a primary and a secondary (alternate) field, either of which can be matched to the target. This is for MACRO fields such as CITY that may have a City or community name different from the post office City name.

CHAR_LEFT_RIGHT

The reference has a left and a right fields, either of which can be matched to the target. This is for MACRO fields such as CITY that may be different for the left and right blockfaces of a street. String similarity measures may be used here too.

NUMBER_SINGLE

The reference and target match on a single number.

NUMBER_INTERVAL

The target number must fall between the two numbers (*from* and *to*) in the reference.

NUMBER_INTERVAL_LEFT_RIGHT

The reference has four numbers, a from-to interval on the left and one on the right. The target number must fall between either one of the intervals. This is the most common of the NUMBER comparison types used in postal address geocoding. This comparison type may also include a similarity comparison (transpositions only).

POSTAL_SINGLE

Reference and target match letter for letter on a single postal/zip code field.

POSTAL_SPLIT

This is the comparison type used when the reference splits the postal code into two separate fields (eg zip, zip4 or fsa, ldu).

POSTAL_LEFT_RIGHT

The comparison type used for postal code with both left and right fields, either of which can be matchted to the target. This is the most common POSTAL type used in postal address geocoding.

POSTAL_LEFT_RIGHT_SPLIT

This comparison type combines the previous two.

REDIRECTION

PAGC uses a redirection strategy to deal with attributes that may appear in the user address standardization or even in the reference standardization which were not part of the original, unstandardized reference schema.

Consider, for instance, the Statistics Canada schema, which has a "Type" field, which does not distinguish between SUFTYP and PRETYP. If the program sends all values to a SUFTYP attribute, it will fail to match with an address in which the type is standardized as a PRETYP.

The strategy is to redirect a non-schema direction attribute to a schema direction attribute, if one exists. Otherwise it is redirected to the (required to be present) STREET attribute. The same method is used for type attributes. The QUALIF attribute is redirected to the STREET attribute. These attributes do not have associated with them a pair of match/mismatch values (See Default Matching Weights). They are classified internally as a NO_COMPARISON comparison type and are scored, using a similarity measure, with the weights assigned to the attribute to which they are redirected.

STANDARDIZATION REFERENCE

The standardizer works by taking the words, phrases and abbreviations in its input and classifying them. Input strings go through a lexical scanner that makes an initial tokenization on the basis of form, identifying ordinals, fractions, numbers, words etc. Each is then looked up in the lexicon and gazeteer and if found there are given whatever definitions and standardizations that correspond with the lookup key. If a string is not found it retains its initial tokenization and input form. The various classifications of the input are what you might term "tokenization candidates". Each possible tokenization candidate, as a string of input tokens (See Input Tokens) is examined. The examination for each entails building a clause-tree and retrieving all the rules of a permissable class (as per a transition table mapping rule types to state) from the Aho-Corasick matrix. Because Aho-Corasick finds (through failure functions) not just rules that match a particular string of tokens, but rules that match a suffix of that string, the tree is built

backwards, from the end of the string forward. Each series of rules that subsumes the address will generate a standardization candidate, which is an ordered string of words or phrases classified by the output tokens (See Postal Attributes). A maximum of six standardizations are retained for the purpose of building or matching.

In applying the rules certain assumptions are incorporated into the transition table referred to above. These are assumptions about how the clause types, each governed by a different type of rules (See Rule Types), fit together. In particular, it is assumed that no clause will come between the house number (governed by the CIVIC_C rule type) and other MICRO attributes (governed by the ARC_C rule type). It is assumed that the CIVIC_C type always precedes the ARC_C type. It is also assumed that extra attributes - the attributes not used for geocoding and governed by the EXTRA_C rule type, will occur either before the house number or after the MICRO attributes. It is also assumed that the MICRO and MACRO (See MICRO and MACRO) attributes can be separated. Although all of these assumptions seem reasonable for Canada and the United States, they may be completely invalid for some other parts of the world.

The Postal Attributes

The standardizer classifies its standardized output by what are called here **Postal Attributes**. The number associated with each class (given here as the token number) appears in the rules. The postal attributes used fall into the functional classes described in the following sections. The standardizer parses the address in both the reference and target using these attributes in order to facilitate matching. The attributes used correspond closely to the SADS classification.

The MICRO/MACRO Attribute Division

Because North American road network files combine in a single record the block faces on both sides of the street, it is convenient to divide the attributes into those which are always the same for both block faces (MICRO) and those which *may* be different (MACRO). That is, MACRO attributes belong to the contiguous polygons rather than the arc itself. This bifurcation necessitates separate standardizations in both the build and match phases.

MICRO attributes.

HOUSE

(token number "1"). (SADS element: "COMPLETE ADDRESS NUMBER"). This is the civic address number. Example, the **3715** in **3715 TENTH AVENUE WEST**. In reference records it is associated with the blockface address ranges.

STREET

(token number "5"). (SADS element: "STREET NAME"). This is the root street name, stripped of directional or type modifiers. Example, the **TENTH** in **3715 WEST TENTH AVENUE**

SUFDIR

(token number "7"). (SADS element: "POST-DIRECTIONAL"). A directional modifier that follows the street name. Example, the **WEST** in **3715 TENTH AVENUE WEST**

PREDIR

(token number "2"). (SAD element: "PRE-DIRECTIONAL"). A directional modifier that precedes the street name. Example, the **WEST** in **3715 TENTH AVENUE WEST**

PRETYP

(token number "4"). (SADS element: "PREFIX TYPE"). A street type preceding the root street name. Example, the **HIGHWAY** in **3715 HIGHWAY 99**.

SUFTYP

(token number "6"). (SADS element: "POST TYPE"). A street type following the root street name. Example, the **AVENUE** in **3715 WEST TENTH AVENUE**.

QUALIF

(token number "3"). (combines SADS elements "PRE-MODIFIER" and "POST-MODIFIER"). Example, the **OLD** in **3715 OLD HIGHWAY 99**

MACRO attributes (SADS element "PLACE STATE ZIP")

CITY (SADS element "PLACE NAME")

(token number "10"). Example "Albany"

STATE

(token number "11"). Example "NY".

NATION

(token number "12"). This attribute is not used in most reference files.

POSTAL

(token number "13"). (SADS elements "ZIP CODE" , "PLUS 4"). This attribute is used for both the US Zip and the Canadian Postal Codes.

EXTRA ATTRIBUTES

These attributes isolated by the standardizer but not used in matching with reference addresses, which are currently assumed to be street addresses:

BLDNG

(token number "0"). Unparsed building identifiers and types.

BOXH

(token number "14"). The **BOX** in **BOX 3B**

BOXT

(token number "15"). The **3B** in **BOX 3B**

RR

(token number "8"). The **RR** in **RR 7**

UNITH

(token number "16"). The **APT** in **APT 3B**

UNITT

(token number "17"). The **3B** in **APT 3B**

UNKNWN

(token number "9"). An otherwise unclassified output.

Output Tokens Ordered by Number

0	BLDNG
1	HOUSE
2	PREDIR
3	QUALIF
4	PRETYP
5	STREET

6	SUFTYP
7	SUFDIR
8	RR
9	UNKNWN
10	CITY
11	PROV
12	NATION
13	POSTAL
14	BOXH
15	BOXT
16	UNITH
17	UNITT

Input Tokens

The tokenizer classifies standardizer input into the following classes. The number associated with each class appears in lexicon (See Standardization Files) entries (which are essentially pre-classified input) and in the rules.

Form-based Input Tokens

AMPERS

(13). The ampersand (&) is frequently used to abbreviate the word "and".

DASH

(9). A punctuation character.

DOUBLE

(21). A sequence of two letters. Often used as identifiers.

FRACT

(25). Fractions are sometimes used in civic numbers or unit numbers.

MIXED

(23). An alphanumeric string that contains both letters and digits. Used for identifiers.

NUMBER

(0). A string of digits.

ORD

(15). Representations such as First or 1st. Often used in street names. Ordinals in **PAGC** are standardized as numbers.

SINGLE

(18). A single letter.

WORD

(1). A word is a string of letters of arbitrary length. A single letter can be both a SINGLE and a WORD.

Function-based Input Tokens

BOXH

(14). Words used to denote post office boxes. For example **Box** or **PO Box**.

BUILDH

(19). Words used to denote buildings or building complexes, usually as a prefix. For example **Tower** in **Tower 7A**.

BUILDT

(24). Words and abbreviations used to denote buildings or building complexes, usually as a suffix. For example, **Shopping Centre**.

DIRECT

(22). Words used to denote directions, for example **North**. Directions in **PAGC** are standardized as a full word (rather than an abbreviation).

MILE

(20). Words used to denote milepost addresses.

ROAD

(6). Words and abbreviations used to denote highways and roads. Exampe: the **Interstate** in **Interstate 5**

RR

(8). Words and abbreviations used to denote rural routes. **RR**.

TYPE

(2). Words and abbreviation used to denote street typess. For example, **ST** or **AVE**.

UNITH

(16). Words and abbreviation used to denote internal subaddresses. For example, **APT** or **UNIT**.

Postal Type Input Tokens

QUINT

(28). A 5 digit number. Identifies a Zip Code

QUAD

(29). A 4 digit number. Identifies ZIP4.

PCH

(27). A 3 character sequence of letter number letter. Identifies an FSA, the first 3 characters of a Canadian postal code.

PCT

(26). A 3 character sequence of number letter number. Identifies an LDU, the last 3 characters of a Canadian postal code.

Stopwords

STOPWORDS combine with WORDS. In rules a string of multiple WORDs and STOPWORDs will be represented by a single WORD token.

STOPWORD

(7). A word with low lexical significance, that can be omitted in parsing. For example, **THE**.

Input Tokens by Number

0	NUMBER
1	WORD
2	TYPE
3	QUALIF
6	ROAD
7	STOPWORD
9	DASH
13	AMPERS
15	ORD
18	SINGLE
19	BUILDH
20	MILE
21	DOUBLE
22	DIRECT
23	MIXED
24	BUILDT
25	FRACT
26	PCT
27	PCH
28	QUINT
29	QUAD

The Standardization Files

In order to function, **PAGC** requires three files in addition to those identified on the command line. These files must reside in a place that the program can find them, either in the same directory as the reference shapeset, the current working directory, or the default installation directory. These files are:

1. The file **rules.txt**, a user-modifiable text file containing the rules used by the standardizer.
2. **gazeteer.csv**, a user-modifiable text file of comma separated values that contains proper place names used by the standardizer.
3. **lexicon.csv**, a user-modifiable text file of comma separated values that contains abbreviations, types and common address words used by the standardizer.

Rule Records

The standardization rules are contained in the file **rules.txt**. It is read in at initialization and stored in such a way that the Aho-Corasick algorithm can be applied to place a string of input tokens (See Input Tokens) in one-to-one correspondence with a string of output tokens (See Postal Attributes).

If **rules.txt** is not found, after the program looks first in the reference shapeset's directory, then the current working directory and lastly the default installation directory, the program will report "Could not find file: rules.txt" and abort.

The rule file consists of a list of rules, expressed as lines of space delimited integers. Each rule consists of a set of non-negative integers representing input tokens, terminated by a -1, followed by an equal number of non-negative integers representing postal attributes, terminated by a -1, followed by an integer representing a rule type, followed by an integer representing the rank of the rule. The rules are ranked from 0 (lowest) to 17 (highest).

The file is terminated by a -1.

The following is an example rule:

2 0 2 22 3 -1 5 5 6 7 3 -1 2 6

This rule maps the sequence of input tokens *TYPE NUMBER TYPE DIRECT QUALIF* to the output sequence *STREET STREET SUFTYP SUFDIR QUALIF*. The rule is an ARC_C rule of rank 6.

Rule types .

MACRO_C

(token number = "0"). The class of rules for parsing MACRO clauses.

MICRO_C

(token number = "1"). The class of rules for parsing full MICRO clauses (ie ARC_C plus CIVIC_C). These rules are not used in the build phase.

ARC_C

(token number = "2"). The class of rules for parsing MICRO clauses, excluding the HOUSE attribute.

CIVIC_C

(token number = "3"). The class of rules for parsing the HOUSE attribute.

EXTRA_C

(token number = "4"). The class of rules for parsing EXTRA attributes - attributes excluded from geocoding. These rules are not used in the build phase.

The rule file is intended to be user-modifiable (See Changing the Rules). Entries can be deleted, added or altered on a provisional basis. The distribution form of the file is not intended to be definitive. The program, when searching for **rules.txt**, will look for it first in the reference shapeset's directory. Modified versions of these files can be placed there in order to supercede other versions of the files.

Lexicon and Gazeteer records

The files **lexicon.csv** and **gazeteer.csv** are read in when the standardizer is initialized. They are used to classify alphanumeric input and associate that input with (a) input tokens (See Input Tokens) and (b) standardized representations.

If one of these files is not found, after the program looks first in the reference shapeset directory, then the current working directory and lastly the default installation directory, the program will report "Could not find file: FILE_NAME" and abort.

The format of these records is that of a comma separated (or comma delimited) file. There are four fields, each of the first three terminated by a comma, and the fourth terminated by the line end. The first field is the definition number. It should be a postive integer. It is used for reference to the lookup value. The second field is the lookup key - the text of the word, abbreviation or phrase that may occur in input. The third field is the input token number, and the fourth field is the text of the standardization value. For example, the lookup key **ST** has these values in the lexicon: **"1","ST",2,"STREET"** and **"2","ST",7,"SAINT"**. The 2 in the first entry indicates that when "ST" is standardized as "STREET", it is classified as an input token of 2, (TYPE). The 7 in the second entry indicates that when standardized as "SAINT", the key "ST" is classified as a STOPWORD.

These files are intended to be user-modifiable (See Changing the Lexicons. Entries can be deleted, added or altered on a provisional basis. The distribution forms of the files are not intended to be definitive. Modified versions of these files can be placed in the reference shapeset directory in order to supercede other versions.

The Statistics File

If the **-z** is specified with **-b** flag, a file is produced giving the hit frequency for each build rule applied to a standardization candidate, and the frequency for which it was chosen as the best standardization. These statistics may prove useful in creating or reweighting

rules for the reference locale. The format of the statistics report is similar to that given for the standardization test (See Standardization Test).

The following are two examples of the statistics from the build of **whatcom.dbf**:

```
Rule 4 is of type 2 (ARC)
: Input : |1 (WORD)||2 (TYPE)|
Output: |5 (STREET)||6 (SUFTYP)|
rank 13 ( 0.825000): hit frequency: 0.118579, best
frequency: 0.811918
9075 hits out of 76531, best 9061 out of 11160
```

This entry is interpreted as follows:

first line

The first line gives the rule number (4) and number (2) and name (ARC_C) of the rule type (Rule Types),

second line

The second line gives the rank (13) - a number between 0 and 17 - and then the value at which this rule is applied in this context (0.825000). The hit frequency (0.118579) is the percentage of times that this rule was tested against the input against the total tests against input, and the best frequency (0.811918) is the percentage of times this rule was selected as best for a reference record.

third line

The third line gives the numbers from which the hit and best frequency in line 2 were calculated.

This rule, then, is the one used to standardize 81% of Whatcom's reference records. A second rule is responsible for another 11%:

```
Rule 694 is of type 2 (ARC)
: Input : |22 (DIRECT)||1 (WORD)||2 (TYPE)|
Output: |2 (PREDIR)||5 (STREET)||6 (SUFTYP)|
rank 12 ( 0.800000): hit frequency: 0.018581, best
frequency: 0.111828
1422 hits out of 76531, best 1248 out of 11160
```

TUNING THE STANDARDIZER

A standardization can be incorrect, in terms of correctly parsing and representing an address form, and yet be acceptable in terms of matching. As long as the reference records and user records can be properly linked, it doesn't much matter how those records are expressed.

Standardization becomes truly problematic when

1. the standardization that would serve as a linkage to the right reference record isn't produced.
2. a reference record cannot be standardized at all

The first problem won't become evident until the match phase, whereas the second problem is usually discovered during the build phase. The process in dealing with them, however, is the similar. The easiest remedy often is to make modifications to your reference or user data. However, this remedy only suffices for small-scale, transient problems. The other solution is to change the behaviour of the standardizer. This is done by changing the standardization files.

If the standardizer produces a problem standardization or fails to produce a standardization at all, and if editing the relevant user or reference attribute records isn't feasible, then we have two principle options:

1. Adding, deleting or changing rules (See Changing the Rules).
2. Adding, deleting or changing lexical entries (See Changing the Lexicons).

Testing Standardizations

To test the standardizer, invoke the utility **pagc_stand** from the command line. Enter the MICRO and MACRO portions of the address as prompted and the best standardization will be printed - or you will learn that the program can't standardize it:

```
[your_name@localhost whatcom]$ pagc_stand
Standardization test. Type "exit" to quit:
MICRO:123 Ta Ta Lost Dog Rd
MACRO:Anywhere BC V0V 0V0
No standardization of MICRO 123 Ta Ta Lost Dog Rd
```

For each of the MICRO and MACRO portions of the address the raw input tokenization candidates and raw standardizations will be printed, followed by the rule statistics for the combined address. However, if **PAGC** fails to find a standardization for either the MICRO or the MACRO, it will not produce the rule statistics.

Raw input tokenization candidates

For each position in the input the associated standardized text is printed along with the associated input token, number and name.

Raw standardizations

For each of the successful standardizations the score is given and then the content. The content will consist of the position in the input, the input token (number and name) selected, the standardized text, and the output token (postal attribute) assigned, number and name.

Rule Statistics

The rule statistics will include all the rules used to form a successful standardization candidate. For each rule the following data is given: the rule number, the rule type, the rank, how that rank is weighted, how many times it was hit out of the total of rules hit.

Example of **-t -v** output.

```
MICRO:123 Ta Ta Lost Dog Rd
MACRO:Anywhere BC V0V 0V0
Input tokenization candidates:
        (0) std: ANYWHERE, tok: 1 (WORD)
        (1) std: BRITISH COLUMBIA, tok: 11 (PROV)
        (1) std: BRITISH COLUMBIA, tok: 1 (WORD)
        (1) std: BRITISH COLUMBIA, tok: 6 (ROAD)
        (2) std: V0V, tok: 27 (PCH)
        (2) std: V0V, tok: 23 (MIXED)
        (3) std: 0V0, tok: 26 (PCT)
        (3) std: 0V0, tok: 23 (MIXED)
Raw standardization 1 with score 0.950000:
        (0) Input 1 (WORD) text ANYWHERE mapped to output
10 (CITY)
        (1) Input 11 (PROV) text BRITISH COLUMBIA mapped to
output 11 (PROV)
        (2) Input 27 (PCH) text V0V mapped to output 13
(POSTAL)
        (3) Input 26 (PCT) text 0V0 mapped to output 13
(POSTAL)
Raw standardization 2 with score 0.675000:
        (0) Input 1 (WORD) text ANYWHERE mapped to output
10 (CITY)
        (1) Input 1 (WORD) text BRITISH COLUMBIA mapped to
output 10 (CITY)
        (2) Input 27 (PCH) text V0V mapped to output 13
(POSTAL)
        (3) Input 26 (PCT) text 0V0 mapped to output 13
(POSTAL)
Standardization of an address has failed
Input tokenization candidates:
        (0) std: 123, tok: 0 (NUMBER)
        (1) std: TA, tok: 21 (DOUBLE)
        (2) std: TA, tok: 21 (DOUBLE)
        (3) std: LOST, tok: 1 (WORD)
        (4) std: DOG, tok: 1 (WORD)
```

```
        (5) std: ROAD, tok: 2 (TYPE)
No standardization of MICRO 123 Ta Ta Lost Dog Rd
```

See The Statistics File for details on the contents of rule statistics output.

The information that can be gleaned from this output can be put to use in correcting standardizations.

In the above case, we can tell from the tokenization and by consulting the lexicon, that the only lexicon entry being applied is:

```
"1","RD",2,"ROAD"
```

Because the words LOST and DOG are not in the lexicon, they will be interpreted as WORD input tokens. TA, however, because it is only 2 letters long, will be interpreted as a DOUBLE.

Checking the rules, we find that there is no rule to translate DOUBLE DOUBLE WORD TYPE to STREET STREET STREET SUFTYPE. This done by searching (using your text editor's find option or grep or some other search method) for the input string.

Recall that a string of WORD tokens in the same field will compress into one:

```
DOUBLE DOUBLE WORD TYPE.
```

Looking up the token numbers, the following input token sequence is constructed:

```
21 21 1 2 -1
```

The negative one terminates the input tokens. This is the string we search for.

Searching, we discover that there is no such rule.

So, we have two options: add *TA TA* to the lexicon or add a new rule to the file rules.txt.

Changing the Lexicon or Gazeteer

If you find an abbreviation, word or phrase that is not being recognized by the standardizer, check the Lexicon and Gazeteer to see if an entry exists there. If not you can add it. (See the format for lexicon entries). If it is not being applied, you can change, or if it is interferring with a correct standardization, you can delete it. Be aware that any alteration may have consequences in terms of other standardizations. If you wish to restrict the scope of your changes to a particular reference, place the files in the same directory as the reference shapeset before building.

As an example, consider the phrase "FS RD". This is used in British Columbia to denote "FOREST SERVICE ROAD" and is used as a type (TYPE). To add this to the lexicon, we look for the lookup key *FS RD*. Not finding any standardizations of this key, we add an entry:

```
"1,","FS RD",2,"FOREST SERVICE ROAD"
```

The 1 is the definition number, the FS RD is the lookup key, the 2 is the input token (TYPE=2) and "FOREST SERVICE ROAD" is the standardization that will be used.

It is advisable to edit these files with a text editor. Some spreadsheets that handle comma delimited files will misrepresent some values.

The precedence in applying input tokens to rules is established not by the definition number but by the order of loading. The Gazeteer is loaded before the Lexicon. The first standardization/input token pair that occurs in the file for a given lookup key will be the first to which the rules are applied. The files are arranged in order of lookup key for the user's convenience. Except for the standardization order for the same lookup key, this has no effect on the rule application.

Another example of a Lexicon change. Let's suppose we want to add "TA TA" to **lexicon.csv**.

Scroll to the appropriate place in the file. The file is in alphabetical order to facilitate editing.

Before the entry:

```
"1","SW",22,"SOUTHWEST"
"1","SWP",1,"SWAMP"
"1","TANK TRAIL",2,"TANK TRAIL"
"1","TEN",1,"10"
"2","TEN",0,"10"
```

There is no previous entry for "TA TA", so give it the definition number "1". The lookup key, the phrase we want the standardizer to recognize, is "TA TA". This should be recognized as a word, so (looking up word in the input tokens (See Input Tokens) and finding that the token number is 1), give it the value. The standardization, the phrase we want the standardizer to emit when it find the lookup key, is the same as the lookup key, "TA TA".

After the entry:

```
"1","SW",22,"SOUTHWEST"
"1","SWP",1,"SWAMP"
```

```
"1","TA TA",1,"TA TA"
"1","TANK TRAIL",2,"TANK TRAIL"
"1","TEN",1,"10"
"2","TEN",0,"10"
```

Save the file in the location you prefer and exit. If you want to make this change a global one, save it to the PAGC installation directory (usually `/usr/local/share/pagc/` - you may need to be su to do this). If you want to keep it specific to shapesets in a particular directory, save it to that directory. Or, if you want it to be applied every time you invoke **PAGC** from a particular working directory (and there is no other copy in the reference shapeset directory you work on), save it in the working directory.

Changing the Rules File

It may be that an address does not standardize properly because of a missing rule. After using the **-t -v** switches on the address *123 Ta Ta Lost Dog Road* (See the testing example) we see that it is tokenized as NUMBER DOUBLE DOUBLE WORD WORD TYPE. NUMBER will be handled by a CIVIC_C rule, so we need a rule DOUBLE DOUBLE WORD WORD TYPE ARC_C rule. We look for DOUBLE DOUBLE WORD TYPE (the two WORD tokens are treated as one by the standardizer) in the rules and find there isn't one.

The format of a rule is described below (See Rule Records). To construct the rule, we take the input token segment we constructed to search for the missing rules (See the testing example) and use it to construct the rule:

```
21 21 1 2 -1
```

The rest of the rule consists of the output segment, type and rank.

The output tokens (See Postal Attributes) we want the input segment to be mapped to are :

```
STREET STREET STREET SUFTYP.
```

Looking up the token numbers, the following output token sequence is constructed:

```
5 5 5 6 -1.
```

The number for an ARC_C rule type is 2. The rank can be (arbitrarily) assigned a value of 9. The complete rule therefore is:

```
21 21 1 2 -1 5 5 5 6 -1 2 9
```

This rule establishes the requisite mapping of input tokens to output tokens in a rule of type ARC_C with an arbitrarily assigned rank of 9. Because there are no rules with the same input tokens with which this rule will collide, the rank isn't too important - any rank at all would result in a standardization. However, you don't want it to be too high in case it gets applied in some future standardization in the wrong context and overrides

Note that in the above example there are two words in the input, but only one word input token in the rule. Multiple WORD/STOPWORD sequences that map to the same output token are represented in the rules by a single WORD token.

One thing to consider is that often a single rule will not be sufficient. You may need to add similar rules to cover the presence or absence of (to mention the most probable attributes) a PREDIR or SUFTYP.

The rule file included with the distribution was generated by a C program. The source of this program, **genrule.c** should be included with the distribution along with the source of another program, named **collide.c**. The collide program takes an amended **rules.txt** file and produces the C header file **gamma.h** and a report **collide.txt**. If the number of rules in **rules.txt** is increased, **PAGC** must be recompiled with the **gamma.h** header. The contents of **collide.txt** give a list of those rules in the file each which take the same string of input tokens but emit a different string of postal attributes.
