Using the Burrows Wheeler Transform for PPM compression without escapes

Peter Fenwick
Department of Computer Science
The University of Auckland
Auckland, New Zealand

p.fenwick@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

PPM (Prediction by Partial Matching)

- PPM considers each symbol in the light of its preceding context.
- PPM builds a context history and can restrict the possible symbols and thereby encode them in few bits.
- PPM learns as it goes, building symbol models from the preceding text.
- Each context might lead into an unknown symbol— emit special `escape` symbol
- Handling escapes is a major problem with PPM.
- Every context must allow for an escape of unknown probability.
- We eliminate escapes by determining all possible contexts and their symbols, transmitting this information as part of the compressed data.

Burrows Wheeler compression

We take an unconventional view of the Burrows Wheeler transform.

- The Burrows Wheeler transform is not itself a compression algorithm .
- The BW transform is a method of analysing the context structure of the input, representing this information in a compact and convenient form.
- The inverse transform can recover all the original contexts from the permuted string (but usually select only that for the original input)
- Usually, comparisons may proceed to the full length of the input.
- We can limit the comparison length to say 4, for order 4 contexts.
- Applying the reverse transform from each position yields all contexts of order 4 with exact coding models for each context.
- No escapes are needed.

Forward and recovered contexts

sym-bol	context		Index	sym-bol	link	Recovered Contexts	
s	sissippi	*mi*	1	s	5	issippi	*mi*
m	ississip	*pi*	2	m	7	issip	*pi*
p	pimissis	*si*	3	p	10	is	*si*
s	sippimis	*si*	4	s	11	is	*si*
i	ssissipp	*im*	5	i	2	issipp	*im*
p	imississ	*ip*	6	p	3	iss	*ip*
i	mississi	*pp*	7	i	6	issi	*pp*
s	issippim	*is*	8	s	1	issipp	*is*
s	ippimiss	*is*	9	s	4	iss	*is*
i	ssippimi	*ss*	10	i	8	issippimi	*ss*
i	ppimissi	*ss*	11	i	9	si	*ss*

Preparing the PPM contexts

- Do standard BW sort, but to say only 4 places (for order 4)
- If contexts are equal, sort on the *following* symbol
- Emit the permuted data (standard BW) as a context description
- Generate the context (4 symbols) for every permuted symbol
- Collect the context/symbol pairs in data structure for PPM coding

Emitting the PPM code

- For order *n*, emit the first *n* symbols in plaintext
- Then, for every symbol emit the symbol according to frequencies of the symbols in its context, but only if the context is non-deterministic, with more than 1 symbol
- Advance the context by 1 position and repeat from 2 above.

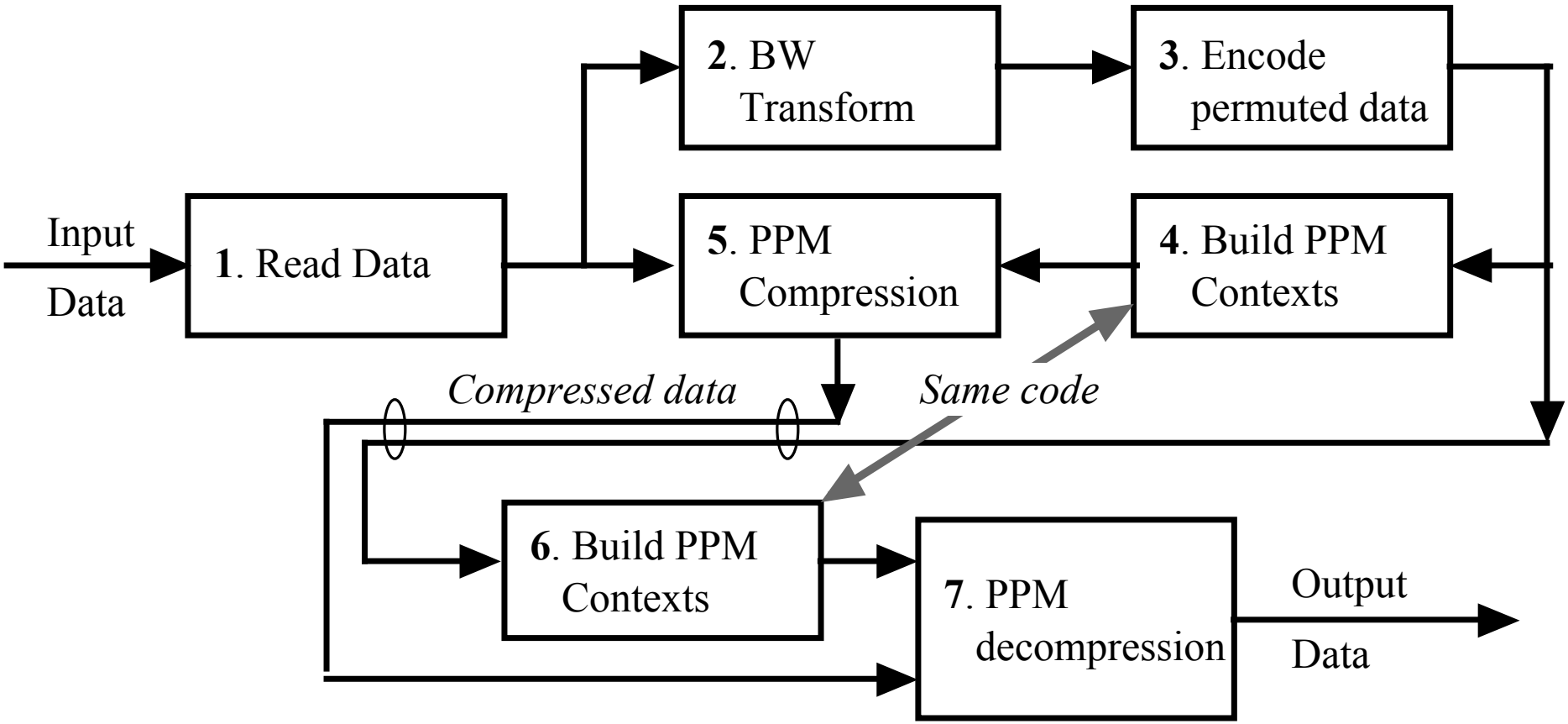
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Constant order PPM compression

context	choice	
		emit *mi*
mi	*s*	
is	*s*	
ss	*i*	
si	*p*,*s*	emit *s*
is	*s*	
ss	*i*	
si	*p*	
ip	*p*	
pp	*i*	
pi	–	

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Data flow of Compressor/decompressor

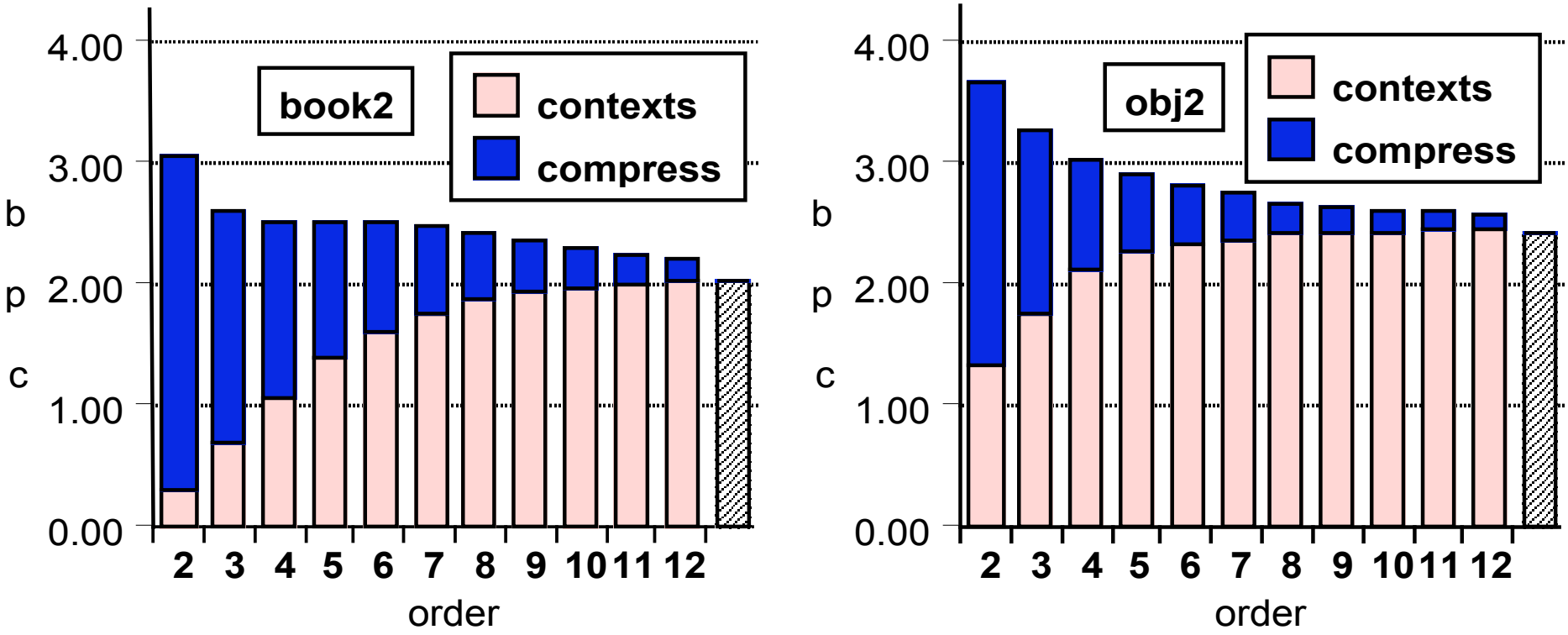


Contexts within Calgary corpus

File	size (bytes)	Order 2	3	4	5	6	7
bib	111,261	1,531	8,155	19,087	27,258	34,273	40,536
book1	768,771	1,826	13,298	49,960	80,781	108,817	135,160
book2	610,856	3,099	18,205	51,792	79,072	103,711	126,590
geo	102,400	13,908	33,268	58,602	67,873	70,936	72,649
news	377,109	4,310	26,952	69,768	101,078	127,535	150,744
obj1	21,504	4,914	8,967	11,419	12,704	13,686	14,458
obj2	246,814	12,040	35,483	57,376	72,681	86,177	98,621
paper1	53,161	1,556	6,155	12,842	18,008	22,361	26,131
paper2	82,199	1,340	5,880	14,794	21,860	27,976	33,357
pic	513,216	3,006	16,987	35,317	42,027	45,960	49,162
progc	39,611	1,746	5,982	11,195	15,115	18,382	21,151
progl	71,646	1,199	4,877	10,450	14,940	19,008	22,681
progp	49,379	1,454	4,805	8,571	11,583	14,269	16,681
trans	93,695	1,990	7,056	12,804	17,508	21,815	25,813

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Compression of two Calgary files



Right hand bar shows typical BW compression

Possible developments – 1

- Compression improves with *shorter* BW contexts and *longer* PPM contexts
- Many order 2 BW contexts can generate longer PPM contexts — can we take advantage of this?
- We may need to encode some secondary context descriptions to fill in the gaps where longer contexts are not generated

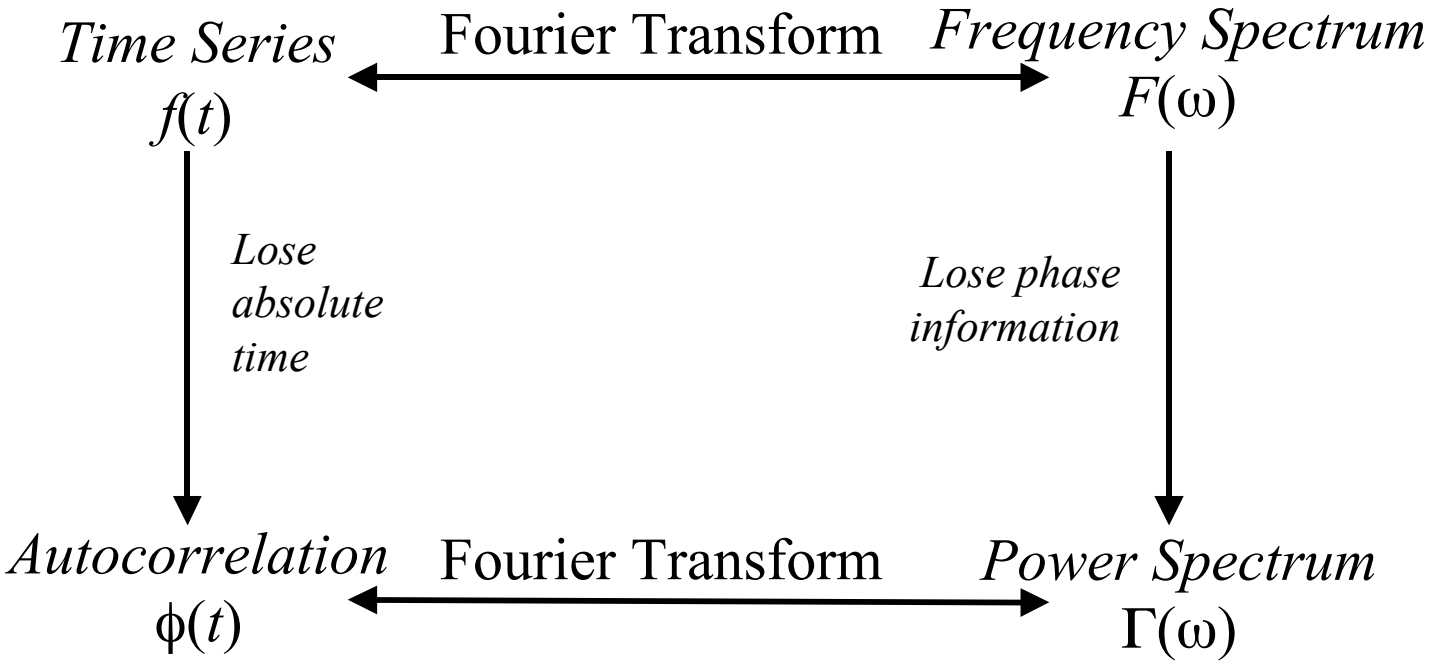
Possible developments – 2

- With the context order always known, is it necessary to include all of the symbols?
- Perhaps some can be omitted, either on a regular basis or marked by a special deletion code (which may be more frequent)?
- This means that we treat some of the encoding/decoding process as an *erasure channel* with some "received" symbols marked as unknown.
- Decoding is then done by a combination of forward and backward searching, as with a Viterbi algorithm for trellis codes.

More thoughts on BW compression

- In mathematics and physics a *transform* converts data between two spaces which provide a complementary view of the problem.
- A well known example is the Fourier transform to convert between *time space* and *frequency space*.
- The Burrows Wheeler transform converts a sequence of symbols between their natural order (*source space*) and context order (*context space*).
- Can operations in one space suggest other operations in the other space?

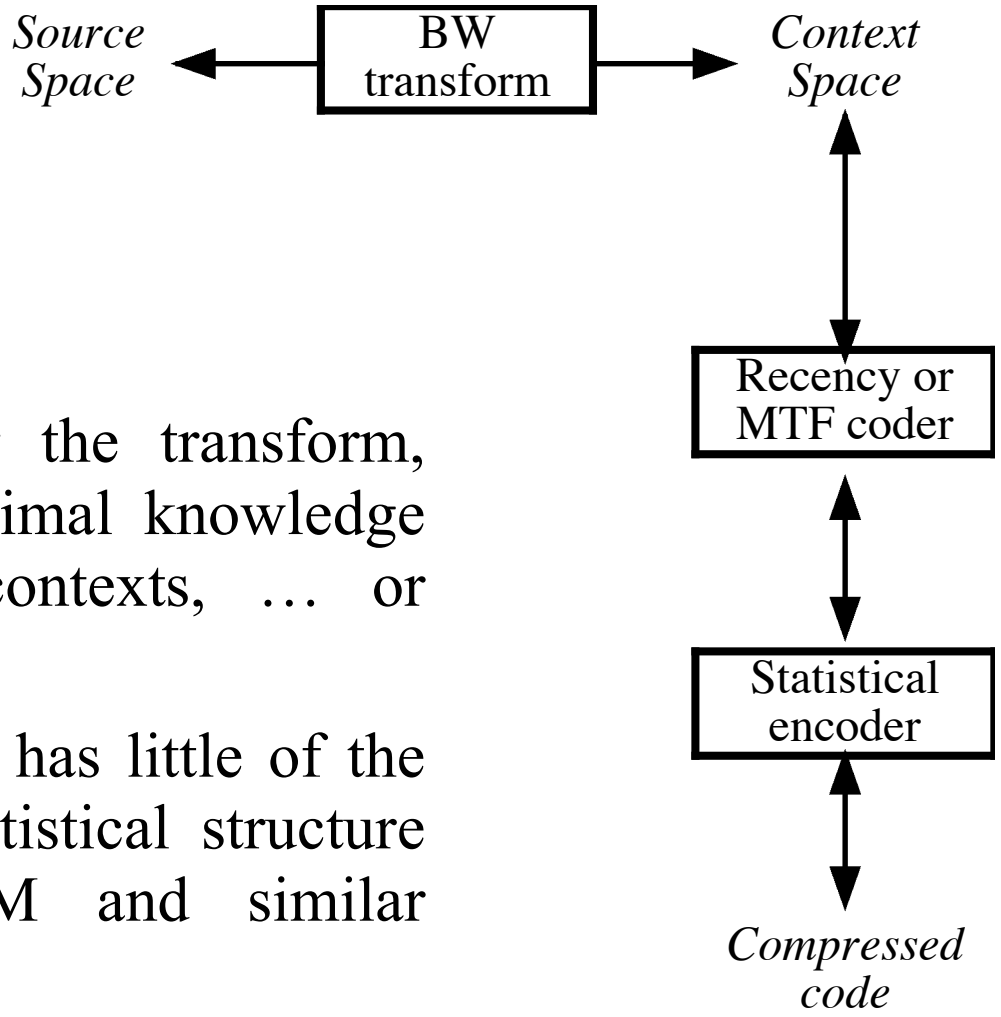
Fourier Transform



- Note overall symmetry
- Reversible horizontally
- Not reversible vertically

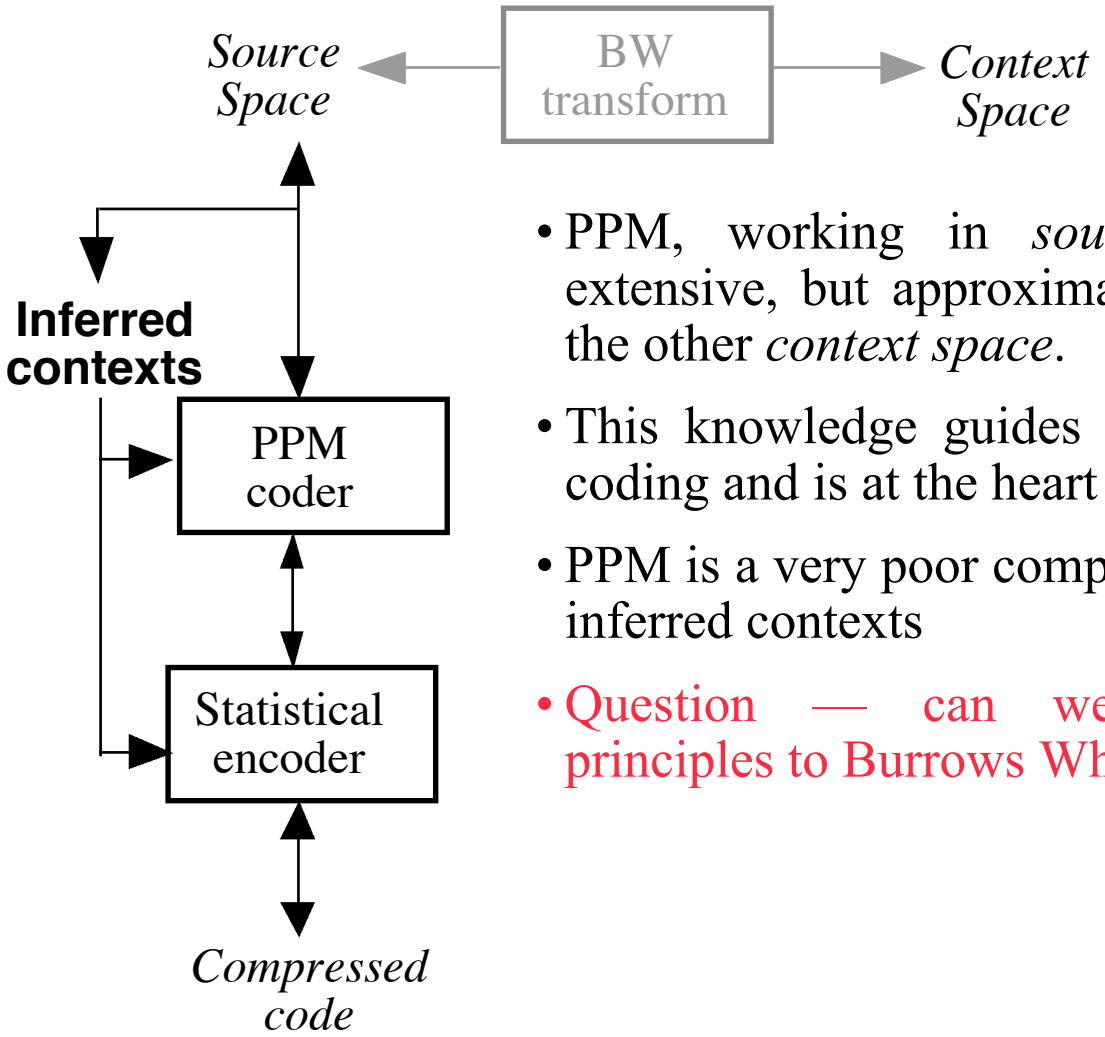
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Burrows Wheeler Compression



- The coding, after the transform, operates with minimal knowledge of the source, contexts, … or anything.
- The context space has little of the "conventional" statistical structure assumed by PPM and similar compressors.

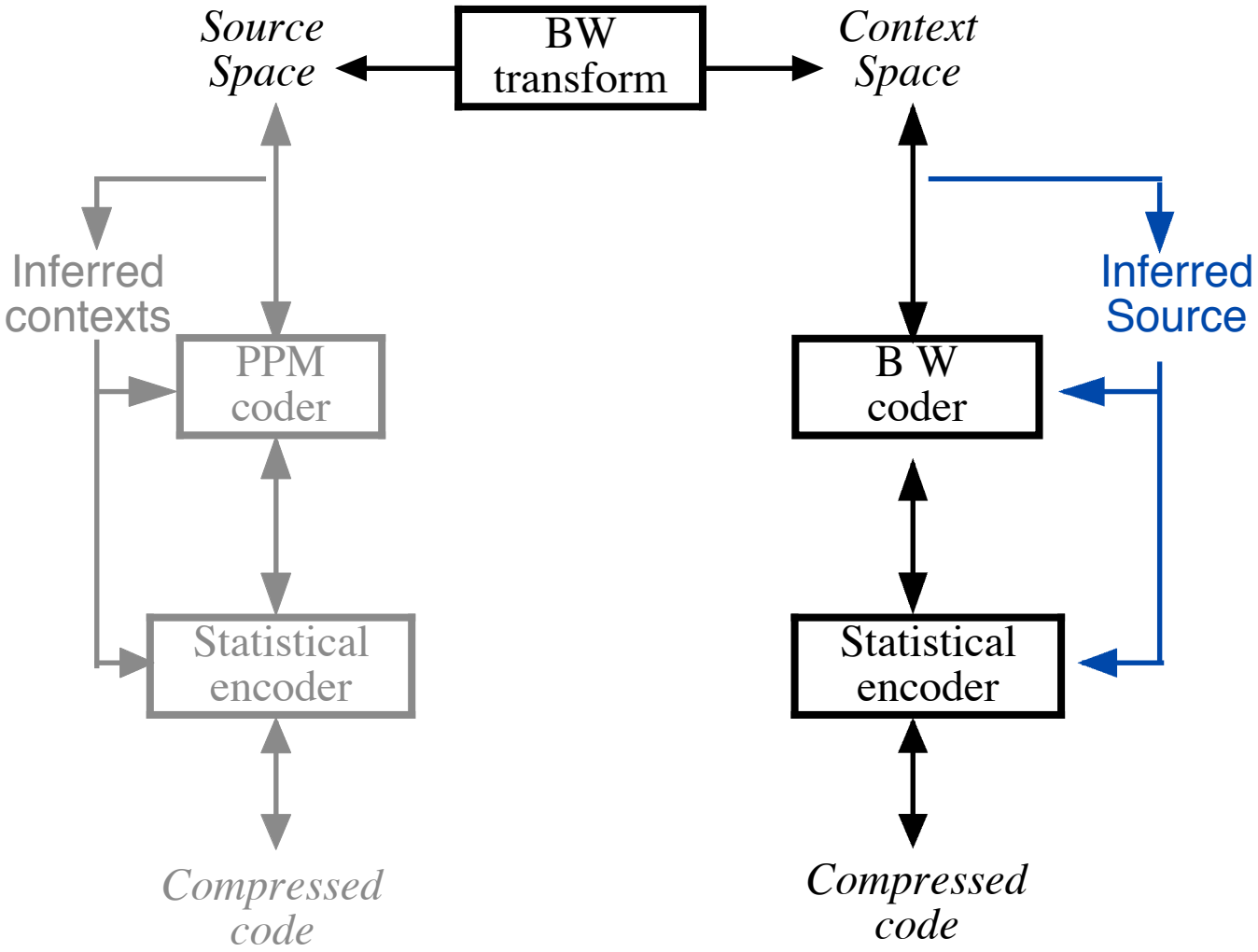
PPM Compression



- PPM, working in *source space* gains extensive, but approximate, knowledge of the other *context space*.
- This knowledge guides both steps of the coding and is at the heart of PPM.
- PPM is a very poor compressor without the inferred contexts
- Question — can we apply similar principles to Burrows Wheeler?

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Modified BW Compression



Deriving the approximate source

Use the reverse transform to build an approximate source

- Count the symbols and transmit the encoded counts (equivalent to the *first* stage of the reverse transform).
- As each symbol (in context order) is processed, link it to its neighbour (as in the *second* stage of the reverse transform).
- At first these links just generate an order 1 context, (which is already known).
- Eventually the links coalesce and we start building longer contexts and can predict the probable symbols.
- As with PPM, we have escapes and the zero-frequency problem, but PPM experience may be a useful guide.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

VERY Preliminary Results

- Use only order-1 contexts – don't develop higher order contexts at all
- Binary and text files may need different compressor tuning.

	New		BW2000
bib	2.188		1.926
book1	2.557		2.356
book2	2.245		1.012
geo	5.272		4.268
news	2.775		2.464
obj1	5.127		3.765
obj2	2.953		2.433
paper1	2.759		2.439
paper2	2.666		2.387
pic	0.892		0.753
progc	2.853		2.476
progl	1.994		1.697
progp	2.044		1.702
trans	1.831		1.488

Average = 2.724 bpc (2.298 bpc)

THE END