Vol. 21 No. 1　　情報処理学会論文誌　　Jan. 1980

英語論文の清書における英単語の自動分節に関する1統計的方法†

浅倉　秀三††

英文の印字において，一般に1行当りの文字数はある数に決められる．その制限にかかる単語は，分節され，ハイフンでつながれる．
本論文は，この分節し，ハイフンでつなぐことを計算機に自動的に行わせる1統計的方法について述べる．
任意の n 文字列の $(n-1)$ 番目の文字と n 番目の文字の間での分節が正しい，正しくないをそれぞれ 1,0 で表す n 文字列論理値遷移行列を導入した．$n=2\sim6$ のそれぞれの場合について，辞書の分節された見出し語を調査し，その行列を作成した．それらの行列において値が1の要素が全要素に占める割合は，n の増加に対し激減した．それで，計算機にそれらの行列の全要素を記憶させることをやめ，値が1の要素と対応する n 文字列を表の形で記憶させた．従来 $n\geq4$ の統計的方法は実用的でないと言われていたが，これによって工学上比較的容易に実現できた．この表を参照する方法で論文を清書する実験を行った．
その結果，筆者の評価方法では $n=4$ 又は5が実用的であることや統計的方法は人間の能力の 60% 以上の能力があることが判明した．

1. まえがき

計算機の端末装置を介して英文のファイルを作り，計算機を利用してそのファイル中の語句の訂正や削除又はそこへ新たな語句の挿入を行う編集プログラムやそれを定められた形式で出力する清書プログラムが実用化されるようになった[1)~5)]．

英文の印字において，一般に1行当りの文字数はある数に決められる．その制限にかかる単語は，分節され，ハイフンでつながれる（以後，この2操作をまとめて分節と呼ぶ）．

清書プログラムは，単語の分節に関する情報を一般には持っていないのでそれができず，その単語を次の行へ送る．しかし，その行の右端に残った空白を，その行の中のほかの空白部分へ分配して行の右端をそろえることができる．これを右端の調整と呼ぶ[4)]（**図1**）．

このようにすることで，ある行の空白部分は，ほかの行のそれと比べ，大であるという問題が生じる．そこで，これを防ぎ，少しでも人間の行うものに近づけたいという希望から，その単語を自動分節する方法が示された[2),4)]．

Thompson, Garland[2)] は，単語の文字列中に分節の位置を示す記号を挿入して分節に必要な情報を清書プログラムに与える方法を示した．

井田[4)] は，分節の規則を清書プログラムに付加する方法を示した．

しかし，自動分節に関する種々な方法に対する検討はまだ行われていないようである．それらの方法は次の3つに大別される．

（1） 分節の規則をプログラムで表す，プログラム書下しの方法．

（2） 分節した単語の表を持ち，それを参照する辞書引きの方法．

（3） 分節した単語から統計的法則を得て，それを利用する統計的方法．

（1）の方法には，その規則は発音と関連があり文字列上ではあいまいな点が多いこと[14)]や，そのプログラムの手続部とデータ部の区別が判然とせずプログラムの保守に困難な点があることなどの問題がある．（2）の方法には，語尾変化した単語の収容方法や辞書引きの方法などに問題がある．しかし，記憶に必要な容量や処理に要する時間が制限されないなら，それらの問題は比較的容易に解決され，その方法は人間の場合に最も近い結果を生むと考えられる．（3）の方法は，つづり誤りの検出・訂正に用いられる統計的方法を修正し，応用することによって実現できると考えられるが，実際にはいまだに行われていないようである．その原因は，おおよそ次の2つと考えられる．

（1） 自動分節に応用する方法が確立されていない．

（2） つづり誤りの検出・訂正に関する統計的方法

† A Statistical Method on Auto Syllabication of English Word for RUNOFF Program by SYUZO ASAKURA (Faculty of Engineering, Chubu Institute of Technology).

†† 中部工業大学工学部自然系列

The purpose of this paper is to propose a statistical method on auto syllabication and hyphenation of the last word of line.

For this method, to use context of n-gram which consists of the (n-1) letters sequence preceding the position to syllabicate the word into two blocks and hyphenate them and one letter following the position is considered. And for realization of this the modified binary n-gram matrix where "1" and "0" indicate that it is right and wrong respectively to do it at that position is introduced.

The 2~6-grams of syllabicated 16,858 lemmata of a dictionary were investigated and binary n-gram matrices were constructed.

The statistical method which utilizes these matrices was examined, where papers concerning natural language processing were used.

It is concluded that this method at n=4 or 5 is practical and the boundary of statistical method's ability is estimated to be above 60% of human's.

(a) 例1　人間によるタイプ印刷

(a) Ex. 1 Human typing.

The purpose of this paper is to propose a statistical method on auto syllabication and hyphenation of the last word of line.

For this method, to use context of n-gram which consists of the (n-1) letters sequence preceding the position to syllabicate the word into two blocks and hyphenate them and one letter following the position is considered. And for realization of this the modified binary n-gram matrix where "1" and "0" indicate that it is right and wrong respectively to do it at that position is introduced.

The 2~6-grams of syllabicated 16,858 lemmata of a dictionary were investigated and binary n-gram matrices were constructed.

The statistical method which utilizes these matrices was examined, where papers concerning natural language processing were used.

It is concluded that this method at n=4 or 5 is practical and the boundary of statistical method's ability is estimated to be above 60% of human's.

(b) 例2　分節しない清書プログラムによる清書

(b) Ex. 2 A part of the output of the RUNOFF program not practising the auto syllabication and hyphenation.

図1 右端の調整

Fig. 1 Arrangement of the right side.

の能力は，一方では英文字列の2文字列遷移行列を使って実験したときのやや低いと考えられる場合[6]があり，他方では単語中の第 i, j, k 番目の位置の文字を対象として3文字列論理値遷移行列 T_{ijk} を使って実験したときのかなり高いと考えられる場合[9]もある．したがって，自動分節に応用した場合の統計的方法の能力や実用性が不明である．

そこで，自動分節に関する統計的方法を検討し，実験してその能力を明らかにすることは実用上から興味ある問題である．

本論文では，2～6 文字列の文脈を利用した統計的自動分節の1方法について述べる．

n 文字列の文脈（本論文では，特に英文字列中のマルコフ性）に関しては，文献 10)～12) を参照されたい．

2. 自動分節に関する統計的方法

2.1 n 文字列論理値遷移行列

n 文字列遷移行列は，つづりの文脈を統計的に表すことができる．したがって，判読し難い文字の推定やつづり誤りの検出・訂正にしばしば用いられてきた[6)~9)]．一般に，その行列の行，列はそれぞれ先行する $(n-1)$ 文字列，n 番目の文字と対応する．その1要素は1 n 文字列と対応する．要素の値はその n 文字列の頻度である．

その行列の要素の値を頻度ではなく，その n 文字列の有，無によって単に 1，0 とする n 文字列論理値遷移行列は，記憶に必要な容量がその行列の場合と比べ，少なくて済むことと計算機処理に適することから利用されてきた[7)~9)]．ただし，この行列でも $n \geqq 4$ になると，これらの長所も少なくなると指摘されていた[9)]．

本論文では自動分節に用いる目的で，任意の n 文字列の $(n-1)$ 番目の文字と n 番目の文字の間での分節が正しい，正しくないをそれぞれ 1，0 で表す n 文字列論理値遷移行列を導入する．

ここで正しいとは，辞書のすべての見出し語において，任意の n 文字列は，それの先頭の文字から $(n-1)$ 番目の文字までの連続する文字のそれぞれの間の関係は音節* 内の文字の連接でも音節の切れ目でもよいが，それの $(n-1)$ 番目の文字と n 番目の文字の間で必ず分節されていることを，正しくないとは，その間で分節されない場合が1例でもあることを意味する．

これは誤った分節をしないための条件である．その必要性を簡単に $n=2$ の場合で示す．辞書の見出し語

* その前後には切れ目が感じられ，それ自身の中には切れ目の感じられない音声の連続における単位を「音節」と呼ぶ[13]が，本論文では，それと対応する一連の文字列を「音節」と呼ぶ．

が次のようにある.

⋮
or
or-ange
⋮
range
⋮

2文字列 ra には, r-a* のときと ra のときがある. 自動分節処理中に2文字列 ra が現れ, それらの文字の間で分節しようとするとき, 仮に r-a の文脈によって分節すると, その ra が range の一部分であるときに誤る. このような誤りを防ぐために ra の間での分節は正しくないとする.

2.2 細かい分節への改善とその限界

2.1 の2文字列 ra に対して文脈の情報を増し, $n=3$ にすると ra は3文字列␣ra** と or-a になり, range の r と a の間で誤って分節することなく, or-ange の r と a の間で正しく分節する. したがって, n を大とすることはこのような細かい分節への改善に導くと期待できる.

しかし, n を大としてもこのような改善が得られぬ例を次に示す. 辞書の見出し語が次のようにある.

⋮
tel-e-gram
tel-e-graph
⋮
te-leg-ra-phy
⋮

2文字列 el, le, eg, gr, ap のそれぞれの文字の間での分節は **2.1** で述べたように正しくないとされる. それで, telegram の le の間での分節は正しくないとされ, 分節は行われない. また, この場合は n を大としても上述のような改善ができない. したがって, n を大としても分節ができないというこの例は, 統計的方法の限界を示唆する.

3. 2〜6 文字列の調査

筆者は, **2.1** で述べた自動分節用の n 文字列論理値遷移行列を $n=2$〜6 の範囲で辞書[13]の分節された 16,858 の見出し語を対象にして作った. そのとき, 単語の境界を示す空白が2以上連続したものを含む n 文字列も含めた***.

それらの 2〜6 文字列論理値遷移行列において値が1の要素が全要素に占める割合を**表1**に示す. その割合は, n の増加に対し激減した. それで, 本論文では計算機にそれらの行列の全要素を記憶させることをやめ, 値が1の要素と対応する n 文字列を表の形で記憶させた*. これによって, 従来 $n\geqq4$ の統計的方法は実用的でないと言われていた[9]が, $n=4$〜6 の場合を工学上比較的容易に実現した.

2〜6 文字列は**表2**のような文字列の型に分類できる. それぞれの型に属す n 文字列の種類の数を**表3**に示す. 表2から, **2.2** で示した n 文字列より $(n+1)$ 文字列の文脈の利用の方が細かい分節を可能にすることの理由は, n が1増加するときすべてが文字の n 文字列からすべてが文字の $(n+1)$ 文字列と先頭は1空白で残りが文字の $(n+1)$ 文字列が作られ, その種類の数が増加することによると考えられる. したがって, 後者の文字列の種類の数と前者のそれとの比は, n の増加における細かい分節への改善の程度を示すと考えられる. その比は, 文字のつながり方に規則性がないときに最大値 27, そのつながり方が唯一に決ま

表1 2〜6 文字列論理値遷移行列において値が1の要素が全要素に占める割合

Table 1 Ratios of 1 number of elements against all number of elements in each n-gram matrix.

n	2	3	4	5	6
割合 (%)	21.2	4.67	0.878	0.064	0.003

表2 2〜6 文字列

Table 2 2〜6-grams.

$L(m)$ \ n	2	3	4	5	6	種類の数
L (2)	*L-L*	*SL-L*	*SSL-L*	*SSSL-L*	*SSSSL-L*	26^2
L (3)	.....	*LL-L*	*SLL-L*	*SSLL-L*	*SSSLL-L*	26^3
L (4)	.....	.....	*LLL-L*	*SLLL-L*	*SSLLL-L*	26^4
L (5)	.....	.....	.....	*LLLL-L*	*SLLLL-L*	26^5
L (6)	.....	.....	.....	.....	*LLLLL-L*	26^6

L: 英字, S: 空白, −: 音節の切れ目,
$L(m)$: 最初から $(n-m)$ 文字列が空白の文字列

表3 2〜6 文字列の種類の数

Table 3 Numbers of each 2〜6-gram.

$L(m)$ \ n	2	3	4	5	6
L (2)	143	15	15	15	15
L (3)	0	837	359	359	359
L (4)	0	0	3797	1774	1774
L (5)	0	0	0	5803	2372
L (6)	0	0	0	0	5045
合 計	143	852	4171	7951	9565

* 記号-は音節の切れ目を表す.
** 記号␣は空白を表す.
*** 例えば, 5文字列では␣␣*ajaka*ɪ なども含めた.

* 付表1を参照されたい.

表 4 $(n+1)$ 文字列中の $L(n+1)$ と $L(n)$ の和と n 文字列中の $L(n)$ の数の比

Table 4 Ratios of sum number of $L(n+1)$ and $L(n)$ in $(n+1)$-gram against number of $L(n)$ in n-gram.

$n \rightarrow n+1$	2→3	3→4	4→5	5→6
比	$\frac{852}{143}=5.96$	$\frac{4156}{837}=4.97$	$\frac{7577}{3797}=2.00$	$\frac{7417}{5803}=1.28$

るときに最小値1となる.

その比を**表4**に示す. この表から, 細かい分節への改善は n が4まではかどり, 5で鈍り, 6で飽和し始めると推定できる.

4. 清書プログラム

本論文の清書プログラムは, 1行当りの文字数の制限にかかる単語に対して分節を試みる. 例えば, その文字列 $a_ia_ja_ka_la_m$ の a_k と a_l の間で分節しようとするとき, a_l に先行する $(n-1)$ 文字列と a_l からなる n 文字列*が **3.** で述べた n 文字列の表中にあるかを確かめる. あれば分節する—この例では $a_ia_ja_k$-をその行に充て, a_la_m を次の行へ送る. 無ければ分節しない. そして, 1文字前へ移りそこでの—この例では a_j と a_k の間での—分節を試みる. 分節が正しい文字の間を見つけるまでこのように順次1文字前へ移って分節を試み, 見つければそこで分節するが, 見つけられなければこれを単語中の該当する文字列が無くなるまで試みる. 無くなれば, 分節を行わない清書プログラムと同様に, その単語を次の行へ送る.

5. 実験結果と検討

筆者は, 自然言語処理に関する論文を分節しない清書プログラム, 2〜6 文字列の文脈を利用した清書プログラム, 人間, それぞれによって清書し, それらの能力を比較した.

印字の規則は通常のタイピングの規則より簡単に次のようにした.

(1) 単語間の空白やピリオド, コンマなどの区切り記号の後の空白は, 右端の調整をする前では1空白とした.

(2) 単語の語頭や語尾の1文字だけの分節は行わない.

(3) 1行当りの文字数を 65 とした.

清書の能力の指標として, 右端の調整をする前の各行の右端に残った空白の総数と印字に要した行の数を数えた(**表5**). 特にその空白の総数は, 次の式によって清書の能力 $E_{(x)}$ を表すものとした.

$$E_{(x)}=\frac{c-x}{c-b}\times 100(\%) \qquad (1)$$

ここで, b, c, x は人間, 分節しない清書プログラム, 任意の清書プログラム, それぞれによる清書の場合のその空白の総数である.

表5のその空白の総数だけを取り出し, **図2**に示す. 実験結果の一部分を**図3**に示す*.

$n=2$, 3 では, それらの n 文字列の表の記憶に必要な容量は少ない. しかし, それぞれ $E_{(x)}=11.5$, 19.7(%)であることやその行の数の減少の程度から実用性は乏しい.

$n=4$ では, これらと比べるとその容量の増加は大きい. しかし, 十分に実用化できる. そして, $E_{(x)}=44.2(\%)$ であることから有用性は非常に大きい.

$n=5$ では, その容量はかなり増加する. しかし,

表 5 実験の結果

Table 5 Result of experiments.

n	分節しない場合	2	3	4	5	6	人間による場合
単語の数	8148	8148	8148	8148	8148	8148	8148
余った空白の総数	2468	2324	2220	1913	1731	1698	1212
行の数	847	846	845	843	839	840	838
n 文字列の数	……	143	852	4171	7951	9565	……

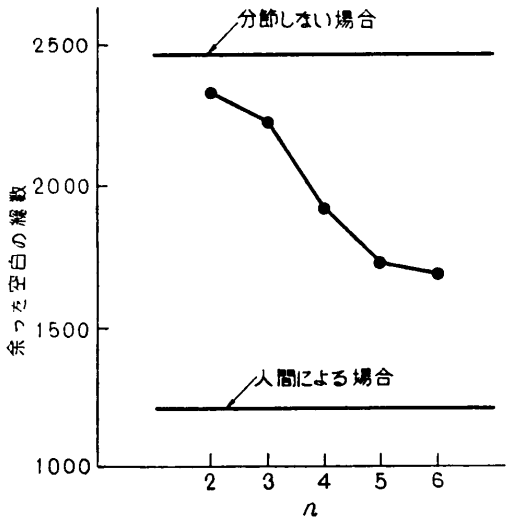


図 2 行の右端に残った空白の総数

Fig. 2 Number of the spaces remained as a function of the n.

* この例では, $n\geq 5$ のとき, n 文字列の先頭から $(n-4)$ 文字列は空白の並びとなる.

* これは本論文のアブストラクトである.

The purpose of this paper is to propose a statistical method on auto syllabication and hyphenation of the last word of line.

For this method, to use context of n-gram which consists of the (n-1) letters sequence preceding the position to syllabicate the word into two blocks and hyphenate them and one letter following the position is considered. And for realization of this the modified binary n-gram matrix where "1" and "0" indicate that it is right and wrong respectively to do it at that position is introduced.

The 2~6-grams of syllabicated 16,858 lemmata of a dictionary were investigated and binary n-gram matrices were constructed.

The statistical method which utilizes these matrices was examined, where papers concerning natural language processing were used.

It is concluded that this method at n=4 or 5 is practical and the boundary of statistical method's ability is estimated to be above 60% of human's.

(a) n=2 又は 3
(a) n=2 or 3

The purpose of this paper is to propose a statistical method on auto syllabication and hyphenation of the last word of line.

For this method, to use context of n-gram which consists of the (n-1) letters sequence preceding the position to syllabicate the word into two blocks and hyphenate them and one letter following the position is considered. And for realization of this the modified binary n-gram matrix where "1" and "0" indicate that it is right and wrong respectively to do it at that position is introduced.

The 2~6-grams of syllabicated 16,858 lemmata of a dictionary were investigated and binary n-gram matrices were constructed.

The statistical method which utilizes these matrices was examined, where papers concerning natural language processing were used.

It is concluded that this method at n=4 or 5 is practical and the boundary of statistical method's ability is estimated to be above 60% of human's.

(b) n=4, 5 又は 6
(b) n=4, 5 or 6

図 3 n 文字列の文脈を利用した清書プログラムによる清書

Fig. 3 Each part of the output of the RUNOFF program using the context of n-gram for the auto syllabication and hyphenation.

$E_{(x)}$=58.7(%) であることやその行の数の減少の程度から有効性は大きい.

n=6 では，その容量はやや増加する．その増加に対し，$E_{(x)}$=61.3(%) であることや逆にその行の数は n=5 のときより増加していることから有効性はやや小さい.

要約すると，本論文の統計的方法の清書の能力の伸びは，n=4 まで順調であり，5で鈍り，6で飽和し始める．これは **3.** で述べた推定と一致する．その行の数の減少は n=5, 6 で飽和する．したがって，利用できる計算機の記憶容量にもよるが，n は少なくとも4，可能なら5とすることが実用的であると結論づける.

自動分節に関する統計的方法は人間の能力の60%以上の能力があると結論づける.

6. むすび

英単語の自動分節に関する1統計的方法を提案し，その有用性を示した.

この統計的方法で最も重要なことは **2.1** で述べた n 文字列論理値遷移行列である．また，**3.** で述べたその行列から作った n 文字列の表の利用は工学上比較的容易に実現可能な n の範囲を拡げた.

本論文では，分節しようとする位置に先行する $(n-1)$ 文字列と続く1文字からなる n 文字列の文脈を利用した．そのほかに，例えばその位置に先行する1文字と続く $(n-1)$ 文字列，その位置に先行する $n/2$ 文字列と続く $n/2$ 文字列，などの n 文字列の文脈を利用することが考えられる．また，分節辞典[14]には語尾変化した単語も分節されて載っているのでそれらも対象として n 文字列を調査することも考えられる．統計的方法の改善に関するこれらのことについては現在検討中である.

謝辞 日頃，御指導頂く慶応義塾大学工学部相磯秀夫教授，御助言頂く電子技術総合研究所言語処理研究室五十嵐実子女史，植村俊亮氏，坂本義行氏，本学鈴木国弘助教授に感謝する．また，データの作成に御協力頂いた中尾照光氏に感謝する.

参考文献

1) Mashey, J. R. and Smith, D. W.: Documentation Tools and Techniques, Proc. 2nd International Conference of Software Engineering, pp. 177-181 (1976).
2) Thompson, C. and Garland, S. J.: DTSS RUNOFF*** Reference Manual, p. 42, Kiewit Computation Center, Dartmouth College (1977).
3) Kernighan, B. W., Lesk, M. E., and Ossanna J. F., Jr.: UNIX Time-Sharing System: Docu-

ment Preparation, Bell Syst. Tech. J., Vol. 57, No. 6, pp. 2115-2135 (1978).
4) 井田哲雄: 英語文書清書プログラム ROFF(2), 東京大学大型計算機センターニュース, Vol. 9, No. 9, pp. 61-65 (1977).
5) 石田晴久: コンピュータによる英語論文の編集・清書法, 東京大学大型計算機センター ニュース, Vol. 10, No. 7・8, pp. 38-41 (1978).
6) Cornew, R. W.: A Statistical Method of Spelling Correction, Inf. & Control, Vol. 12, pp. 79-93 (1968).
7) Thomas, R. B. and Kassler, M.: Character Recognition in Context, Inf. & Control, Vol. 10, pp. 43-64 (1967).
8) Riseman, E. M. and Ehrich, R. W.: Contextual Word Recognition Using Binary Digrams, IEEE Trans. Comput., C-20, pp. 397-403 (1971).
9) Riseman, E. M. and Hanson, A. R.: A Contextual Postprocessing System for Error Correction Using Binary n-Grams, IEEE Trans. Comput., C-23, pp. 480-493 (1974).
10) Shannon, C. E.: Prediction and Entropy of Printed English, Bell Syst. Tech. J., Vol. 30, pp. 50-64 (1951).
11) Newman, E. B. and Gerstman, L. J.: A New Method for Analyzing Printed English, J. Exptl. Psychol., Vol. 44, pp. 114-125 (1952).
12) 浅倉秀三: 英文字列中のマルコフ性の起源とそれが及ぶ範囲について, 電子通信学会論文誌, Vol. 61-D, No. 12, pp. 933-939 (1978).
13) Hornby, A. S. et al.: Idiomatic and Syntactic English Dictionary, p. 1548, Kaitakusha, Tokyo (1972).
14) Takenaka, J.: A Pocket Dictionary of English Syllabication, p. 283, Ai-iku shuppan, Tokyo (1977).
15) 竹林 滋: 英語音素論概説, 現代英語教育講座 4, pp. 1-53, 研究社, 東京(1976).

(昭和54年6月27日受付)
(昭和54年9月20日採録)

付表 1 2文字列の表の全部と 3～6 文字列の表の一部分
($n-1$) 番目の文字と n 番目の文字の間での分節が正しい n 文字列
A. Table 1 Whole list of 2-grams and parts of lists of 3～6-grams.

␣: 空白

(a) $n=2$

bc	bd	bf	bg	bh	bj	bk	bm	bn	bp	bs	bv	bw	cc	cd
dc	df	dh	dj	dk	dm	dn	dp	dq	dv	eh	fc	fh	fn	fp
gw	gz	hb	hc	hd	hf	hg	hh	hk	hn	hp	hq	ih	ii	iw
km	kp	kt	kw	lh	lr	lw	mc	md	mf	mh	ml	mr	mv	mw
mm	np	nq	nr	nv	nw	oh	oj	pb	pc	pd	pg	pk	pm	pw
sf	sg	sj	sr	sv	tb	td	tf	tg	tk	tm	tn	tp	tv	uj
wg	wm	wp	ww	wy	xa	xc	xg	xh	xi	xl	xo	xp	xq	xs
yj	yw	zv												

cm	cn	cq	cz	db										
fs	gb	gd	gf	gp										
kb	kc	kd	kf	kg										
nb	nf	nh	nj	nl										
rj	rw	rz	sb	sd										
uq	uu	vv	wb	wf										
xu	yb	yd	yf	yh										

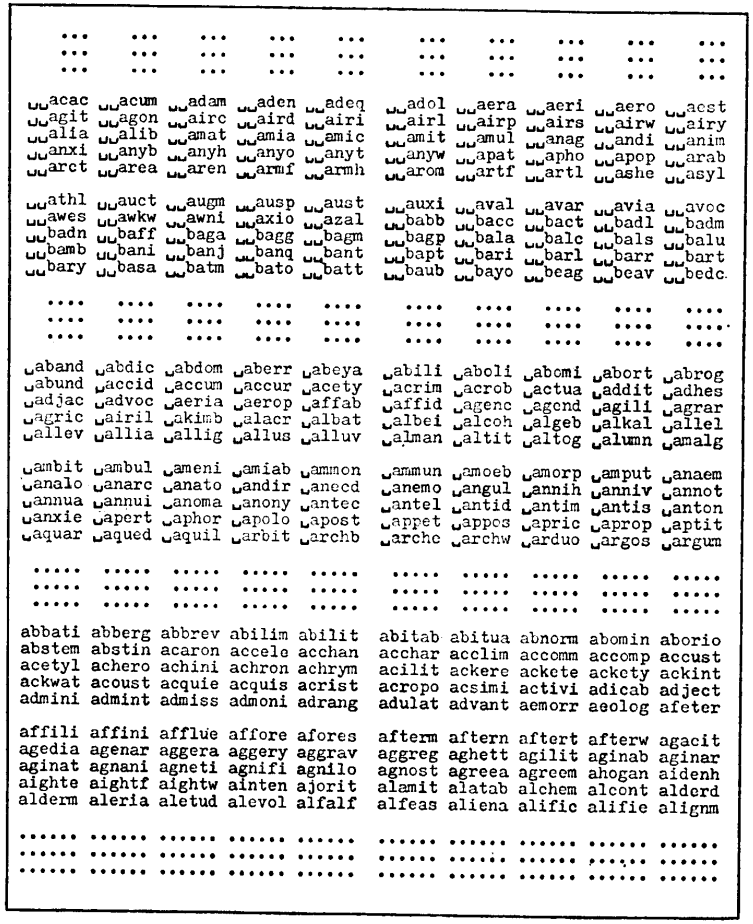
(b) $n=3$

␣aj	␣ak	␣aq	␣ej	␣ia	␣ib	␣io	␣iv	␣og	␣oh	␣oz	␣ub	␣uk	␣uv	␣yc
abb	abd	abh	abj	abn	abs	acc	acm	acq	acs	adb	adc	adf	adh	adj
adm	adn	adp	adq	ads	adv	adw	aea	aeg	aen	aeo	aet	aev	afn	agg
agm	agp	agt	agy	aia	aie	aka	akd	akf	akl	akn	akw	aky	alb	alc
alg	aln	alr	alw	aml	amr	ams	anb	anf	anh	anj	anl	anq	anv	anw
anz	apd	apk	apn	app	apw	arj	arw	arz	asb	atf	atg	atk	atl	atm
aup	avv	awb	awd	awf	awi	awp	awt	awy	axa	axe	axh	axi	axl	axo
axy	ayb	ayf	ayg	ayh	ayi	ayl	aym	ayn	ayp	ayt	ayw	baz	bef	beh
bek	bem	beq	bew	bim	bio	bip	biz	btf	byg	byh	byp	byr	byt	byw
caj	cao	ceb	cef	ceg	chb	chc	chf	chg	chn	chp	chw	cib	ciu	cka
ckb	ckc	ckd	ckg	ckh	cki	ckm	cko	ckp	ckr	ckt	ckw	cky	coe	coh
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

(c) $n=4$

␣␣aj	␣␣ak	␣␣aq	␣␣ej	␣␣ia	␣␣ib	␣␣io	␣␣iv	␣␣og	␣␣oh
␣␣oz	␣␣ub	␣␣uk	␣␣uv	␣␣yc					
␣abb	␣abd	␣abh	␣abj	␣abn	␣abs	␣acc	␣ack	␣acm	␣acq
␣adh	␣adj	␣adm	␣adv	␣aeg	␣aeo	␣aet	␣aff	␣agg	␣agn
␣alb	␣alc	␣ald	␣alf	␣alg	␣alk	␣alp	␣alr	␣als	␣alt
␣alw	␣amb	␣amm	␣amn	␣amp	␣anc	␣ang	␣ani	␣ank	␣ann
␣ans	␣anv	␣any	␣app	␣ara	␣arb	␣ard	␣arg	␣arr	␣ars
␣asb	␣asc	␣atl	␣atm	␣att	␣aub	␣aud	␣aut	␣awf	␣axi
␣axl	␣azu	␣baz	␣bef	␣beh	␣bei	␣bem	␣beq	␣bew	␣bey
␣bia	␣bim	␣bio	␣bip	␣biz	␣byg	␣byp	␣bys	␣byw	␣caj
␣cet	␣cic	␣cid	␣cip	␣coe	␣coh	␣coq	␣cyc	␣cyp	␣dej
␣dio	␣dua	␣dut	␣dyi	␣eag	␣ebo	␣ecc	␣ecs	␣ecz	␣edd
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
abal	abia	abie	abul	abyh	abyr	acab	acac	acao	acea
acef	acel	acha	achf	achr	acip	acka	ackb	ackc	ackd
ackg	ackh	acki	ackm	acko	acks	ackw	acky	acua	acui
acum	acun	acuo	acuu	adab	adat	adeq	adia	adii	adio
adiu	adol	adua	adyl	adys	aede	aels	aeri	aero	aery
aest	afeg	afet	afty	agab	agam	agar	agaz	agem	agia
agit	agod	agul	ahli	aide	aiet	aila	ailc	aile	aili
aill	ailr	ailt	ailu	ailw	aina	ainb	aind	aing	aini
ainl	ainm	aino	ainy	aira	airc	airl	airp	airw	airy
aiss	aiti	aitr	ajam	akab	akef	akes	akew	alab	alaw
....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....

(c) $n=6$



(d) $n=5$